André Bernardo Quintanova

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Hybrid System of Distributed Automation

Dissertação para obtenção do Grau de Mestre em Engenharia Eletrotécnica e de Computadores

Orientador: Doutor Luís Filipe Figueira Brito Palma, Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

	Júri:
Presidente:	Doutor João Francisco Alves Martins
Arguente:	Doutor João Almeida das Rosas
Vogal:	Doutor Luís Filipe Figueira de Brito Palma

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2016

Hybrid System of Distributed Automation

I dedicate this to my parents, my hamster and bacon

Acknowledgments

Firstly I want to thank my parents for always motivating and supporting me along this academic stage of my life.

I would like to thank Professor Luis Brito Palma, advisor of my dissertation, for the opportunity, support and for enduring my ramblings.

To my friends Vasco Brito and Afonso Maria for traveling with me through this journey. And to my friend Ana Areias for reviewing my dissertation.

Additionally I would like to thank Tucker Emerson and the Kepware Company for helping me in technical issues and for giving me a yearly license to all of their products.

Abstract

One of the most important tendencies in the development of the industrial automation is the application of intelligent control systems within factories, which focuses heavily on networked architectures. Following this line of thinking, the goal of this dissertation resumes itself in the implementation of a distributed system that controls two physical processes, where the system components not only trade information between each other, but also have that same information be accessible remotely and within HMI equipment.
The controllers were conceptualized to offer different functional modes with high customization available.
This system also takes resource of an OPC server, so it allows, not only the communication between different manufacturer PLC controllers but also the connection with remotes clients
The implemented remote clients hold the intent of demonstrating the versatility of this architecture and are, namely, an operational historian that registers information and a data viewer, which allows the use of more advanced methods of monitoring.

Keywords: Distributed System, Programmable Logic Controllers (PLC), Remote Clients, OPC®, Matlab®, Scilab®.

Resumo

Uma das mais importantes tendências no desenvolvimento da automação industrial é a aplicação de sistemas de controlo inteligentes, no ambiente de manufatura, que focam arquiteturas fortemente baseadas em redes de comunicação. Seguindo esta linha de pensamento, o objetivo desta dissertação resume-se na implementação de um sistema de controlo distribuído de dois processos físicos, onde os componentes não só trocam informação entre si, como essa mesma informação é acessível remotamente e por equipamento HMI.
Os controladores foram implementados de forma a oferecerem diversos modos de funcionamento com alta vertente de customização.
Este sistema também recorre a um servidor OPC de forma a não só possibilitar a comunicação ente controladores de fabricantes diferentes como também a conexão a clientes remotos.
Os Clientes remotos implementados possuem o intuito de demonstrar a versatilidade desta arquitetura e são, nomeadamente, um historiador operacional onde toda a informação é registada e um visualizador de informação, que possibilita a implementação de métodos de monitorização mais avançados.

Palavras-chave: Sistema Distribuído, Controladores Lógicos Programáveis (PLC),Clientes Remotos, OPC®, Matlab®, Scilab®.

Acronyms

AC	Alternating Current
AR	Auto Regressive
ASCII	American Standard Code for Information Interchange
CAN	Control Area Network
CPU	Central Processing Unit
DC	Direct Current
DCS	Distributed Control System
DP	Distributed Periphery
EPA	Enhanced Performance Architecture

EPROM	Erasable Programmable Read Only Memory
FBD	Function Block Diagram
HMI	Human Machine Interface
HZ	Hertz
IL	Instruction List
I/O	Input/Output
IP	Internet Protocol
JMP	Jump
LAN	Local Area Network
LD	Ladder Diagram
LRC	Longitudinal Redundancy Check
OPC	Open Protocol Communication
OSI	Open Systems Interconnection
PA	Process Automation
PAC	Programmable Automation Controllers
PCA	Principal Component Analysis
PI	Proportional Integral Controller

PLC	Programmable Logic Controller
RAM	Random Access Memory
RFID	Radio-Frequency Identification
ROM	Read Only Memory
RPM	Rotation per minute
RTR	Remote Transmission Request
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SFC	Sequential Function Chart
SRR	Substitute Remote Request
ST	Structured Text
SVD	Singular Value Decomposition
TCP	Transmission Control Protocol
USB	Universal Serial Bus
WAN	Wide Area Network

Nomenclature

$\boldsymbol{u}$	Control Signal
$\boldsymbol{y}$	Sensor Value
$\boldsymbol{r}$	Set-point value
$\boldsymbol{e}$	Set point error
ê	Set point error
$\boldsymbol{P(t)}$	Proportional Component
$\boldsymbol{I(t)}$	Integral Component
$\boldsymbol{Kp}$	Proportional Gain
$\boldsymbol{Ki}$	Integral Gain

$\boldsymbol{Ynn}$	Predicted Sensor Value
$\boldsymbol{\sigma^2}$	Control Action Variance
$\boldsymbol{I_{Harris}}$	Normalized Harris Index
$\boldsymbol{S}$	Covariance Matrix
$\boldsymbol{\lambda_i}$	Eigen values
$\boldsymbol{P_i}$	Eigen Vectors
$\boldsymbol{T}$	Score Vector
$\boldsymbol{\theta}$	Model Parameters Vector

Table of Contents

Figure List

List of Tables

1. Introduction

In this chapter, the reasoning that led to the execution of this dissertation is done.

In sub-chapter 1.1 it is mentioned the context to what motivated the implementation of this system.

The goals and contributions of this project are descripted in sub-chapter 1.2 and 1.3, respectively.

Lastly, it is given a notion of how the dissertation is organized, in sub-chapter 1.4.

1.1 Motivation

The tendency of the current industrial automation scene relies heavily in the concept of Industry 4.0, which illustrates the implementation and further development of smart manufacturing environments based in strong network communications.

This initiative aims to satisfy customer demands for new, high-quality customized products and, at the same time, reduce production time cycles and resource utilization. Such goal is done by not only heavily networking a manufactory process but also by applying more flexible, open and smart distributed system architectures.

In many manufacturing companies, the used control systems are based in outdated concepts and methods of production with strong sequential logic and controllers with very specific tasks. Additionally, this systems have very rigid structures with small synchronism and interaction foundations. Therefore, this project intends to propose a reliable and cheap distributed system that holds

many of the benefits the concept of Industry 4.0 has to offer (interoperability, interaction, flexibility, autonomy).

1.2 Goals

Initially, the goals of this dissertation resumed themselves to the detailed study of the most important topics concerning distributed systems for the area of industrial automation and control. From programmable controllers and fault detection methodologies, to communication protocols and human-machine interaction. With this notion, the main goal of this dissertation is to present a possible distributed system with high interoperability, supervision and accessibility features.

The system is conceptualized to hold an architecture composed by a varied array of technologies and operational methodologies, where all the main components communicate with each other and the most important information is compatible and accessible remotely. Different manufacturer programmable controllers are used with the goal of making the system more versatile and compatible. It also intends to conjugate different types of SISO controllers and monitoring techniques, permitting several possible work states and different manners of observing control information.

Concerning the contextualization for the application to which the system was developed, it was in regards to the post treatment of stainless steel parts. This method is named pickling, and it is usually applied to a post welding phase. In short, pickling is essentially a timed treatment where the welded parts are submersed in chemical solutions, with the sole goal of fortifying the weakened sectors due to heat exposure.

With this context in mind, this system's main purpose is to control and monitor the transportation of the part from a prior workstation to the pickling station and its respective chemical treatment procedure.

1.3 Contributions

The main contribution of this project is to present a new concept for industrial systems in an effort to approach the industry 4.0 tendency without it translating in expensive production line restructuration.

In terms of scientific value, this system was centralized in an Open Protocol Communication server, which allows for all of the system's information to be neutral and available to any of the system's components (current or future additions). The differentiation comparatively to other systems, is that typical

distributed systems never use servers and networks as the foundation of the system, usually, they are used just for information collection and for high level monitoring.

It also presents a fully functional industrial prototype within the control laboratory, which allows for a didactic way of illustrating the importance of automation and distributed control, presenting several technologies applied within the industry.

1.4 Thesis Organization

This dissertation is constituted by five chapters, including the introduction and is organized in the following manner:

Chapter 2 - State of Art and Technology

This chapter studies the important topics used within the project in an automation aspect. The first section is dedicated to automation in general and its evolution; the second section approaches the important programmable logic controllers. The third section is dedicated to distributed control systems structures and production methodologies. The fourth section focuses on the industrial networking, going over its concepts and the most used communication protocols. The last section concludes and encompasses all of the former described technologies and makes the connection with the project in the following chapter.

The first four sub-sections also describe current developments that give the reader a notion of what is being researched and developed.

Chapter 3 – Architecture and Implementation

The third chapter is dedicated to describing every aspect of the system, from architectures and control methodologies to hardware structure and software implementation.

It is divided in global architecture and in the detailed overview of both physical process control systems, supervision, communication and remote clients.

Chapter 4 – Experimental Results

On the fourth chapter, the main experimental results and notes concerning the main risk factors within the overall system will be presented.

Chapter 5 – Conclusions

On the final chapter, the conclusion of the dissertation is elaborated and the presentation of possible future works following the automation distributed systems area.

2. State of the Art and Technology

In this chapter, it will be approached and explained with some detail the automation thematic within the project. From the theory behind the subjects to popular alternatives, current projects and scientific research being developed.

The sub-chapter 2.1 will briefly go over what automation is and what categories it can be divided onto, its historical development as well as the industrial revolution and the current projects being worked.

The sub-chapter 2.2 is going to describe the main technology behind the industrial Automation, called Programmable Logic Controller (PLC), from the hardware constitution, used programing tools to current developments.

In the section 2.3, the subject is distributed control systems, where the following topics will be discussed: the concept, comparisons against non-distributed systems, applications, manufacturing methods and current developments.

In sub-chapter 2.4 we will focus on industrial networks, which were crucial for the further development of the industrial control systems. We will see how industrial networks are structured and how the specific protocols developed work, as also what differentiates the industrial networks from the commercial networks, as also current researches about the subject.

The sub-chapter 2.5 concludes and encompasses all of the former described technologies and makes the connection with the project in the following chapter.

2.1 Industrial Automation

The concept of automation is, essentially, the creation and implementation of technology that allows monitoring and control of a device, so it possesses the capability to perform a task on its own. Ultimately, having the objective of improving production efficiency and minimizing task associated human intervention.

Such devices might be classified as machinery, factory processes, heat treating ovens, steering and stabilization mechanisms and much more.

Automation is virtually applied in all of the industrial sectors, from manufacturing (Figure 2.1), transportation, to the better use/reach of utilities, building automation and military purposes.

Figure 2.1 - Tesla Manufacturing line [1].

Its concept has been realistically implemented through means that include mechanical, hydraulic, pneumatic, electronic and microprocessor technologies. In the modern times, typical systems include a combination of all of the mentioned above.

It is worth noting that the idea is proved to be applied since the beginning of humanity, but the term itself was adopted after the creation of the automation department from General Motors, in 1947. It was at this time that the vision of fully automated factories was starting to be very prominent in manufacturing companies, especially after continuous research and development, which proceeded the invention of the first feedback controllers in the mid-1700s [2].

2.1.1 Types of Automation

There are many different types and ways of implementing the automation concept. However, they generally fall in the following three categories (comparison made in Table 1):

- **Fixed/Hard automation**: this category is illustrated by the use of specific automate components to perform very detailed and simple tasks, usually on lines of production that are mainly structured by sequential operations. Its use is often justified when there is the need of high demand of one kind of product to be manufactured [3][5].

The worst factor associated to fixed automation systems is the incapability of adapting, specially, when it is desired to modify the initial product to which the system was designed. For instance, this type of automation can be found in the automobile industry, steel rolling and paper production.

- **Programmable automation**: the production equipment in these kinds of systems is designed to be able to change work states for different product specifications. The operation sequence is coded and programmable, allowing the creation of new configurations and operation states through reprograming [3] [5]. It is associated with this structure, a significantly lower production rate, therefore generally being adopted by factories that have higher priority in product variety comparatively to product demand. For instance, this type of automation can be found in the systems that use numerical controlled machine tools, industrial robots and programmable controllers.

- **Flexible automation**: this structure is an upgraded concept of programmable automation with the goal of having the same benefits but with higher production rates. The main difference lies essentially on the fact that there is no time lost between work state changes, therefore, allowing a much faster and variable production sequence [3], [5].

Table 1 - Comparison between Types of Automation.

	Fixed	**Programmable**	**Flexible**
Production Rate	High	Low	Medium
Cost	Low	Medium	High
Adaptability	Low	High	Medium

2.1.2 The Evolution of Automation

On what concerns automation, it is important to have an insight of how it, developed through the ages into the modern control systems we have today. As observed in the **Figure** 2.2, the automation evolution was heavily influenced by the Industrial Revolution.

Although the general perception with autonomous controllers is that they are related with factory production structures since the mid 1800's, there were already existing inventions in the Ancient Greek and Arabic communities that represented the raw concept of automation and most of them were based in the use of float-valve regulators [5][6]. The basic functioning of these regulators are, when open, it would cause the retained liquid to drop with a periodic frequency directly associated with the percentage of the regulator's opening.

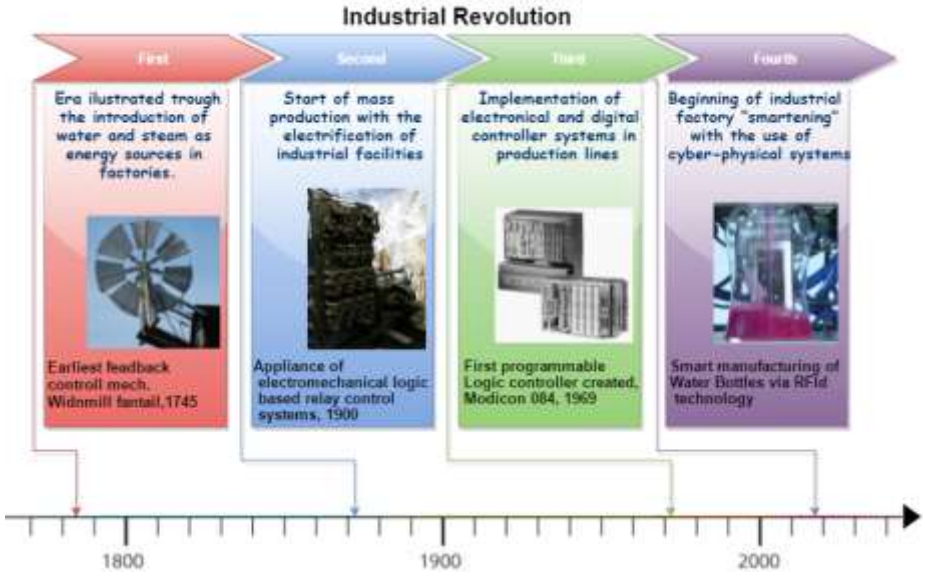


Figure 2.2 - Automation progression timeline.

This almost periodic re-allocation of liquid, made possible the invention of what is believed to be one of the first feedback control devices documented in the history of humanity. That device was the water clock of Ktesibios (Figure 2.3), created in Alexandria, Egypt around 250B.C. [5]. This clock allowed the notion of time to be a quantifiable variable and it was arguably one of the most exact time measuring inventions, until the invention of pendulum clocks in the 17th century [5][6].

Figure 2.3 - Ancient water clock illustration [8].

Patented around 1745 by Edmund Lee, one of the earliest feedback control mechanisms created, was used to manipulate the sails of the windmills, with the purpose of controlling the gap between grinding stones [5][6]. This concept contributed greatly for major advances in control systems around the beginning of the first Industrial Revolution, in the 18th century. In the same time period, as steam and water resources were starting to be used as means of energy, it eventually helped the creation of the first steam engine governor (Figure 2.4), created in 1788 by James Watt.

The steam governor offered a proportional controller that regulated the amount of fuel admitted onto the engine, therefore maintaining a near constant-speed, without providing exact speed control. This mechanism was only considerably acknowledged by scientists when James Maxwell published a paper entitled 'On Governors' (1867) that truly established the start of the theoretical foundation for control theory [7].

Figure 2.4 - An example of a steam governor [9].

Besides the research and development around the steam governor and the theory behind it, the 1800's and early 1900's eras, are mainly illustrated by the creation of simple process task controllers for temperatures, pressures, liquid levels and speed of rotating machines [6].

With the introduction of electrical energy to the factories by the second Industrial Revolution, a new means of automation was heavily adopted, by using logic, based on electromechanical Relays (created in 1835 by Joseph Henry), due to the high demand of controllers in factories and power plants (1900 through 1920) [5].

This logic was used to facilitate the implementation and representation of manufacturing programs based on relays (typically on-off controllers). With this concept, there was also the adoption of central control rooms, where operators observed charts with periodic acquired data and manually actuated on the factory process (trough switches and/or valves).

Even though the evolution in control theory was quite noticeable, mostly via the concept above as well as innovations in the transportation area (use of gyroscopes for ship stabilization and primitive auto pilot systems), there were still a lot of conceptual challenges [5]. Namely, a lot of confusion in the reason why the controllers presented different behaviors, especially when changed the dynamic and environment the processes the controller was designed for.

In 1932, the concept of negative feedback was understood and implemented, clearing a lot of the former challenges at hand. This concept added the ability of precisely impacting the actuators of a process, in order to get the desired results [5] [6].

Conjoined with the advancement of wired/wireless communication systems, this era (1935-1950) denominated "The Classical Period", was considered the main basis for the modern controllers [6].

In the mid-1950s, the third phase of the Industrial Revolution started to be noticed through the creation of data processing machines [5]. These devices made possible the beginning of the implementation of digital controllers, which minimized even more the intervention of a human in a factory process. Until then, the control systems were all predominantly analog based and/or used relay concept structures.

During this period, is it also worth mentioning that there were a lot of advancements in control theory, mainly via the acknowledgment that control systems are non-linear and the existence of significant errors in the sensors caused by noise. There were also new concepts created, such as the use of physical behavior equations and "black box models" [6].

All of these concepts being developed, along the significant fall of prices in digital processor devices, led to the considered official beginning of the third Industrial Revolution, with the creation of the Programmable Logic Controller (Figure 2.5). This device helped minimize significantly the costs associated, mainly to the constant re-calibrating of already existing controllers implemented, by allowing a means of reprogrammable higher logic take place in the factories, and also by offering tools with programing languages that were easily understood and used by their own factory technicians [5].

Figure 2.5 - The first Programmable logic Controller Modicon 084 [10].

This era was mostly translated by evolving the industry through the implementation of electronics and digital controllers.

The current era, denominated by fourth Industrial Revolution or Industry 4.0, is mainly illustrated by bringing communication systems (e.g. internet network technologies) into the factories, with the goal "smartening" the industrial processes. Allowing further efficiency and profit, through the creation of long distance distributed control systems, where each component of the production line, autonomously communicates with each other and adjusts itself to a required functioning state.

This era primarily focuses on the development of concepts that use the following principles [11]:

- **Interoperability**: every main machinery has to be capable to communicate autonomously through the internet;
- **Virtualization**: data bases full of acquired information associated with virtual models of the real production line implemented;
- **Real-time data acquisition;**
- **Decentralized systems**: the ability of each component making decisions on its own;
- **Information accessible without location restrictions**;
- **Flexibility**: easy adaption of each component to different desired work tasks;

Due to most stakeholder's natural inertia into investing on further developing already existing factories, this era is still on its early ages. However, the believed tendency is to see complex networks being more prominent in lines of production in the coming years.

2.1.3 Current State and Developments

The ultimate goal of the global current research is to bring to reality viable and effective implementations of the concept of Industry 4.0; in other words, to create smart factories that are extremely networked, distributed and flexible.

This sub-chapter will approach a few researches that approach the main theme of automation.

On what concerns advancements in interoperable and distributed systems, there is a concept being developed where every material resource and

component of a product has a special RFID tag, as shown in Figure 2.6 [12][13]. When going through the assembly line, those materials are able to communicate with each other, allowing the identification of which materials should go through the same route, in order for a specific product to be created. Furthermore, each assembler robot is integrated with a scanner to read the information from the group of materials and consequently know exactly which manufacturing task to perform. This concept relies greatly on the use of radio frequency identification chips, and it is considered a main future tendency on what concerns factory layout and product planning optimization.

Figure 2.6 - RFID usage on smart manufacturing of bottles [12].

There are also being developed major industrial projecting and implementation tools trough virtual 3D prototyping [14]. These tools rely on the idea that while creating the 3D model of the desired product, at the same time, the program would be able to project the entire production line and machinery associated, offering specific market analysis on the materials needed. This would significantly lower initial placement costs by companies.

Another aspect being researched focuses on human resource advancements. The idea is based on consolidating, in real time, the factory manufacturing operations with the available workers, considering their many individual chrono-biological attributes [15].
The workers would be connected to the network at all time, allowing each workstation to acknowledge which employee is to work next, and adapting itself to any possible limitations or disabilities. It would also create a mean of data acquisition, which would be conveyed into more efficient human resource deployment. The main goal of this concept is to add further flexibility onto a factory.

In a last remark, there is a new tendency that came to fruition with the evolving penetration of 3D printing systems into the market [16]. This notion is denominated additive manufacturing and it is based on the use of a new concept

of “3D printing” implemented into conventional manufacturing lines (Figure 2.7). The concept uses as resource materials that can be presented in the form of fine powder, and what it does is constructing components (simple or complex) layer by layer, contrary to the current concept used of 3D printing that is based on milling techniques.

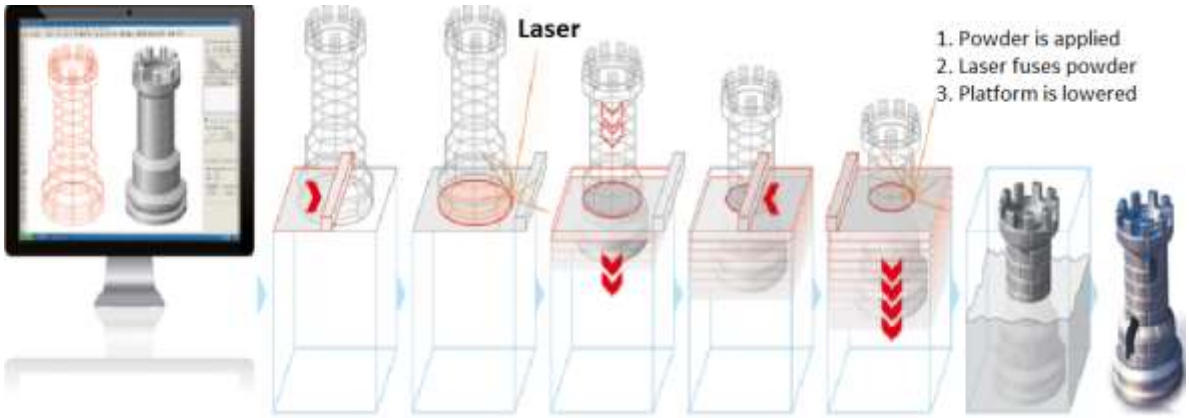


Figure 2.7 - Additive manufacturing principle [16].

2.2 Programmable Logic Controllers

A PLC is a digital device that processes, in real time, the acquired information from an industrial equipment to which is coupled with (typically the ones that resort to electromechanical technology) and consequently manipulates the process into the desired work state (Figure 2.8).

This is done by offering several kinds of digital and analog input/output ports that are compatible with different types of industrial machinery, therefore allowing the reading of sensors and a following calculated actuation on the components of the main process equipment.

Figure 2.8 - A PLC industrial cabinet [20].

As mentioned on the previous chapter, the PLCs were an invention of great significance during the second Industrial Revolution, having the main focus of digitalization of the production environment.

It allowed a new level of flexibility and ease of control by programming and executing logical instructions. The first PLC to be produced was Modicon 084, in 1964.

It is worth noting that this device is still very prominent on the modern industrial scene, largely due to their durability, reliability, cost and ease of technical implementation compared to the more advanced control systems.

2.2.1 Structure and functioning

A PLC normal cycle of work is based on the main following actions (Figure 2.9) [17]:

1. **Input Scan**: processor time slot reserved to access and verify every single input port acknowledged as in use and note their state changes on the input memory table;

2. **Program Scan**: with the information gathered in the former action, the CPU takes the new changed/unchanged information retrieved from the input ports check and executes the respective previously programmed sequence of logic instructions. While the instructions are being executed, the output memory table is kept updated;

3. **Output Scan**: uses the results from the programmed logic that affect output ports by accessing the output memory table and activates/de-activates them by applying or cutting off the hardware designed voltage of each port;

4. **System operations**: last group of operations in a PLC cycle is reserved for more advanced actions such as communications, diagnostics and so forth.

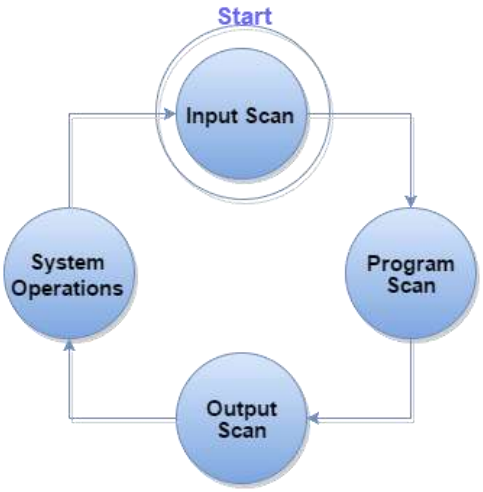


Figure 2.9 - A PLC high level cycle of operation.

Concerning its hardware structure, it is essentially divided in four sectors: input/output modules, central processing unit (CPU), memory and programming terminal (Figure 2.10) [17].

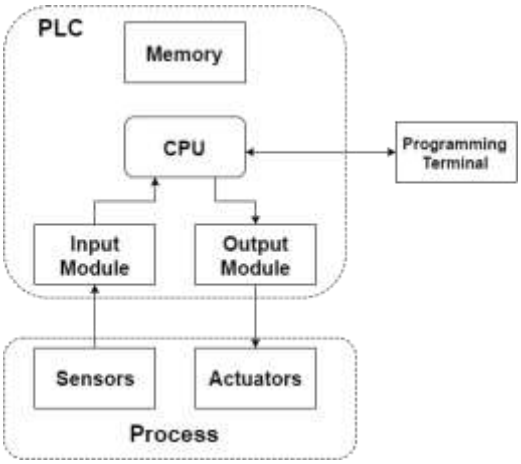


Figure 2.10 - PLC hardware structure.

Going a bit over the components and starting with the central processing unit, all PLCs are generally based on microprocessor structures. Their function is to control and supervise all the operations within the device itself and execute instructions from the programed logic stored in the memory. All the components communicate periodically with the CPU via a main internal bus system. Some more advanced PLCs are built with more than one CPU, in order to achieve efficiently more complex instructions [17].

On what concerns the memory, PLCs contain both random access and read only modules (RAM/ROM). The memories are composed by the following segments (Figure 2.11) [17]:

- **Executive Memory**: this sector of the memory is ROM type, and it is where the main manufacturer default system operations are stored. It instructs the PLC on how to scan I/O modules, interpret user programed logic, test and diagnose internal modules.
- **System Memory**: RAM type sector reserved for results and information of system only operations, such as error codes and module work states.
- **I/O Image Table**: reserved area of the memory that is constantly being updated with the new values read/wrote from the input/output modules Each I/O port has a dedicated memory slot with an individual address. RAM type of memory applied in this case.
- **Data Memory**: it has a similar way of functioning to the previous sector mentioned, but applied to instruction logic values that must be stored, such as values that are calculated, or counters and timers being used. Sometimes it is divided into a constant value and variable value area (RAM).
- **User Program Memory**: portion of memory where the code programmed by the user is allocated, during the CPU work cycle, when the program scan phase is achieved. This is the place from where the CPU takes the operations to execute. Usually comes in RAM format, but can be set in EPROM if desired (a more definitive mean of assuring the information stays allocated).

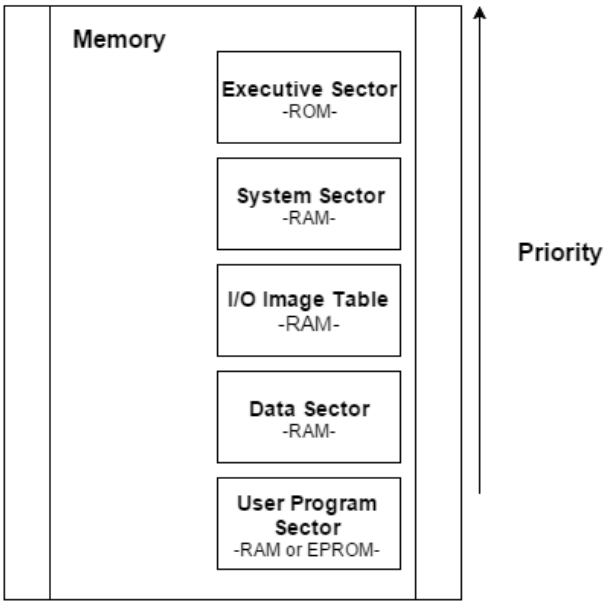


Figure 2.11 - PLC Memory structure.

The programming terminal is what offers the PLC the compatibility to communicate with secondary devices with the intent of being programmed, tested and monitored by the user. The technologies generally offered are through USB variations and several network technologies (Modbus, Serial communication, Ethernet, Etc.)

In regard to the Input/output modules, they are the means to interact with industrial mechanisms that are a part of the control system. Usually, PLCs offer a digital module where the inputs typically work at 24V DC, and the outputs have either a discrete 24V DC functioning or a relay/TRIAC supported output where an output of up to 240V AC/1A can be used. The ports are often electrically isolated to protect the device from any undesired power-supply surges.

2.2.2 Programming Languages

In the implementation of a successful control system using PLCs, there must be a code programed by the user with all necessary operations. In order for that to happen, the PLC offers several programming languages, which initially had the goal of using similar logic and representation to the former techniques applied by the engineers and technicians. In one way, this was one of the biggest factors to the success of such devices, making them easier to implement and modify without the need of specialized technicians and, therefore, more cost effective.

The languages currently supported by the IEC 61131-3 standard for programmable logic controllers are the following:

- Ladder Diagram (LD);
- Function Block Diagram (FBD);
- Instruction List (IL);
- Sequential Function Chart (SFC);
- Structured Text (ST);

On a side note, most of the PLCs are multi task based, allowing the simultaneous use of combinations of different operations with varied languages. It is also worth mentioning that all variables used, need to be specifically allocated on a space of memory conceptually implemented to handle the type of the variable. Next, the summaries of such languages are presented

2.2.2.1 Ladder Diagram

This language was the first one offered by the initial PLCs created and its main goal was to be simple, useful and, most importantly, easily comprehensible by the technicians. It was designed in such a manner that could visually and conceptually illustrate the important relay logic (very dominant before the digitalization of the industrial environment occurred).

As seen in Figure 2.12, the logic in Ladder offers several "branches" and flows from the left to the right. Each branch can be considered a rule that conceptually starts on the verification of one or several inputs and verifies if they satisfy certain conditions. In other words, verifies if the input variables are activated/de-activated and, if so, it proceeds into the operation phase, naturally altering output variables. All the rules typically run simultaneously in a continuous cycle.

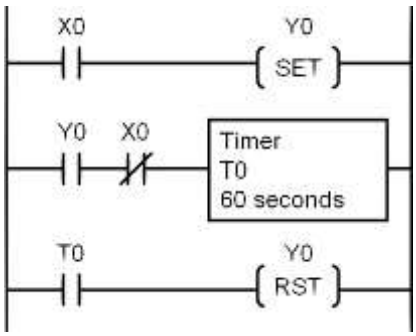


Figure 2.12 - Example of a simple button activated timer in Ladder.

The instructions are made to visually resemble electric wiring. Therefore, the inputs are represented by contacts and the outputs by coils, and both are interpreted as Boolean variables.

This language was conceptualized for tasks that mainly resort to Boolean combinatorial logic, consequently being based mostly in Boolean conditions (e.g..: AND, OR,NOT). Regarding the operations it offers, besides setting a variable "ON/OFF", there is also the possibility of using counter and timer associated actions in form of pre-programed logic blocks [19].

The foundation of this language and style of programming allows for a global and easy way of use and understanding, especially by the less expert technicians. However, its disadvantage is that, when tasks get complicated, the programming complexity and process resources grow exponentially [18]. Moreover, developing simple sequential structures becomes a very complicated task to implement, and when the code becomes significantly large it is hard to interpret and debug, even when well structured.

2.2.2.2 *Function Block Diagram*

The conceptual principle behind the Function Block Diagram (FBD) languages is data flow - the information continuously flows through the code from inputs to the outputs (left to right), meaning that the operations between the transition are all done through function blocks.

The function blocks can be sequentially concatenated by connecting the outputs of the former block to the inputs of a subsequent block, therefore facilitating the creation of more complex instructions. The blocks not only offer Boolean logic and simple arithmetic functions, but also comparisons, mathematical functions (ABS, COS, TAN) and can also have user programmed instructions, simplified to a higher level (Figure 2.13).

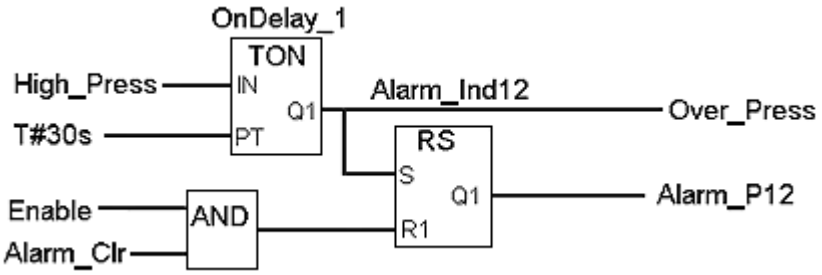


Figure 2.13 - Example of a simple alarm in FBD.

In contrast to the Ladder language, FBD is usually analyzed from the right (output) to the left (input). And it holds advantages compared to the former mentioned language, in the aspect that is easily analyzed and diagnosed by users that are not familiar with electrical schematics [18].

The FBD is a language that deals very well with simple numerical processing tasks, such as limit verification, scaling, and more. The cons regarding this language are that the larger the code, the harder it is to monitor, edit and analyze.

2.2.2.3 *Instruction List*

The Instruction List (IL) language is one of the first textual programming tools offered by PLCs [19]. It is considered to be a low level coding tool, similar to the assembly language, and its foundation follows the Accumulator concept. In other words, it represents a chain of mathematical or logical operations that occur in a stepwise fashion, basing the current operation with the results from the previous action.

In this programing language there are several code lines and in each line there is one single operation to be executed that may be applied to a single or several variables (Figure 2.14).

```
1.1   LD      SM 0.1          On for One Scan
1.2   MOVD  # 4000 , VD200    Put 4000 in address
1.3   LD      SM 0.1          VD200
1.4   MOVW  # 41 , VW10       Put 41 in address VW10
1.5   LD      SM 0.1
1.6   DIV  VW10 , VD200       DIV Value in Address
                              VD200 on the Value VW10
                              put result in the address
                              VW200
```

Figure 2.14 - Example of a conditional allocation in IL.

It supports loop programming and conditional verification trough commands like jump (JMP), which takes the execution pointer to an address where the desired following instruction is, as well as offering several arithmetic and comparative operators.

The main advantages of this language lie in the fact that it is textual and condensed, being executed and processed a lot faster, when compared to graphical languages. Furthermore, no specific editors are needed adding a layer of portability to this language. It is also a language generally preferred by programmers that prioritize performance over diagnosing and monitoring [18].

2.2.2.4 Sequential function Chart

The Sequential Function Chart (SFC) is a more flexible adaptation of the Grafcet language, allowing an easier implementation and debugging check of automation systems with a well-defined sequential evolution.

This language can be divided in a sequential way of coding with two main components: action steps/states and conditional transitions (Figure 2.15).

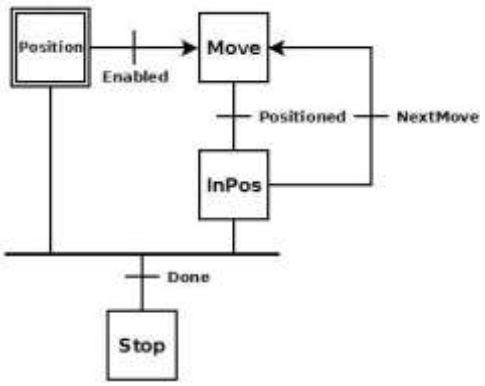


Figure 2.15 - Example of a robot movement routine code via SFC.

The way it works is: an initial state is triggered, assuring the activation of a group of variables, and if the existent condition in the following transition is satisfied, the system will disable the current state and move onto the next state and its individual variable operations. Transitions can activate several parallel states at the same time. There can be more than one initial state when there are different concurrent tasks within the project.

This language allows simpler code execution, due to the fact that there is no need to scan the whole code each cycle, but rather just the current state and the following transition. This fact also allows fast identification of malfunctioning transitions and "dead end" states [19].

There are several disadvantages worth mentioning. First, amongst the graphical languages, SFC is by far the one that consumes more memory resources, making the execution simpler but a lot heavier and slower. It is also the language that lacks the most in portability - every equipment/software associated has its unique configurations, making it impossible to pass a coded SFC from one source to another without using a third party conversion tool or ultimately having to re-code the whole program [18].

2.2.2.5 Structured Text

Structured Text programming language is the last standardized language offered by the majority of the PLCs produced and sold today. It is the tool with the highest level of programming capability of all the mentioned before in this sub-section. It is often compared to a Pascal language further developed into industrial applications [19].

The language is composed of statements separated by semicolons -these statements can be pre-defined or can be user programmed subroutines (Figure 2.16). The coding is composed by various instructions, such as loop iterations, conditional verifications and further advanced mathematical equations. Every variable is defined previously to the main code of instructions it alludes to, and it can be of the type related to the I/O ports of the PLC internally stored memory types or constant values.

```
avg := 0;
i := 0;
WHILE (i < 5) DO
        avg := avg + f[i];
        i := i + 1;
END_WHILE;
avg := avg / 5;
```

Figure 2.16 - Example of an average value calculation code via ST.

Concerning pros and cons, this tool brings for a more productive means of programming in the current day where the complexity of the problems to be solved have grown very significantly. Besides being well-structured, compact and having more advanced instructions, its highly compatible and consumes lower processing resources. The cons usually mentioned of this language are its textual nature, and the fact that it is arguably harder to diagnose and perform maintenance [18].

2.2.3 Current State and Developments

About the advancements of this technology, it is worth noting that PLC devices have existed for over 40 years and their structure gives great emphasis to programming languages that have been surpassed. Furthermore, the great developments in PC performance and the introduction of Programmable Automation Controllers (PAC) in the industrial market have caused the PLC to decrease in market share dominance of the current manufacturing scene.

Although there is not an established consensus on what should a PAC be or what are the exact differences from a PLC, PAC devices are globally seen as highly advanced PLCs - they hold a more open architecture and modular design, that further facilitates interoperability and communication, more robust PCUs that can perform greater complexity routines, highly developed software tools that bring lower complexity and more modern programming languages to the table (C,C++, JAVA) [22].

PACs do seem to be the successor of the PLCs, technologically wise, but it really depends on the complexity of the control system to be implemented. PACs are very expensive when compared to PLCs, and PLC manufacturers, due to the pressure of the PAC devices market penetration, have been implementing notable upgrades to the current PLCs, from additional CPU features, high speed communication capabilities, larger variety of I/O ports to superior performance and efficiency [21].

There is also a notable philosophical shift in the industrial scene today: the customers do not prioritize individual technology performance but instead focus on global system functioning. Consequently, there has been less focus on buying individual components (PLCs, inverters, etc.) and more emphasis on industrial solution services to create the best global system possible for the situation [22].

A brief conclusion to this sub-chapter is that, even though the PLC devices have been surpassed technologically, depending on the industrial solution complexity and funding, they are still used immensely on the current manufacturing control systems.

2.3 Distributed Control Systems

A Distributed Control System (DCS) is a conceptual automation structure that uses individual controllers and monitoring technologies. The system components are placed strategically throughout the whole industrial plant with the goal of processing/executing specific tasks in large and complex processes. Each component of the DCS communicates in a type of hierarchy level organization, via network, for higher command and supervision purposes [23].

This structure can also be seen as a global system composed of varied sub-systems with individual tasks, contrary to the concept of centralized control systems, where all of the tasks and components of a manufacturing line are controlled and processed through only one main controller. Comparing the two of them, DCSs do bring a higher initial investment cost but portray a higher degree of flexibility, simplicity, control reliability and performance, especially in systems that are subjected to production changes or extensions.

The most valued key feature of this concept is redundancy. DCSs are all designed to accommodate multiple elements that control/monitor the same process component, allowing the existence of back up resources in case of anomalies and/or critical failures. Such feature is extremely important in the modern day mass production industrial lines, where there are major security risks and huge profit losses in case of production stoppage [25].

Applications of DCS are seen over all of the industrial areas, especially the ones that require a higher degree of plant complexity and high cost efficiency, such as power generation, transportation fuel refining, pharmaceutical and highly automated food and beverage production.

2.3.1 Production methods

Before continuing the further description of distributed systems, it is important to enunciate the usual methodologies applied to manufacturing systems, in other words, what are the most common strategies for assembling a commercially viable product. This reason is justified due to the fact of DCS being generally associated to manufacturing processes, so it is important to have a notion of the most used production methods.

In the modern days, it is hard to define a specific factory production structure due to the fact that the industrial scene usually uses combinations of different manufacturing methods that best adapt to their personal goal and cost efficiency. With that in mind, there are three main manufacturing concepts: Job, Batch and Flow production [26].

2.3.1.1 Job Production

Job Production, or One-off production, is a production method that illustrates the execution of custom/personalized work. It is a unique and time constrained kind of service that usually serves as a third party contracted production resource.

This method assures that the product/service created matches the customer needs exactly, and it is usually characterized by highly specialized tasks and quality work. Therefore, it is a significantly expensive method, due to the use of highly skilled staff and time demanding activities [27].

Examples of this kind of production go from building new installations, extending a production line, installing new equipment, maintaining/fixing manufacturing processes to producing highly specialized materials for a new upcoming technology.

2.3.1.2 Batch Production

The Batch production concept lies in producing a fixed number of units to satisfy a certain customer's order. It is usually best applied in low-mid initial capital companies that produce several types of the same product.

The way it works is as such: if there are 3 types of units to be produced, the factory is set with the needed specifications for one type and produces all the units needed; afterwards, it is reset for the next type of unit. Initially, the raw materials are picked and passed through sequential manufacturing stages that are divided by different workstations.

This method is mostly applied in companies that have no need or cannot afford continuous production. If there is a sudden cut in demand, this method allows production stoppage without suffering huge profit losses (this factor is best suited for factories that produce seasonal items) [27].

Examples of businesses that generally use this method are in the areas of bakery, clothing, pharmaceutical and construction materials.

One method worth mentioning that is usually associated with batch production is Just in Time manufacturing, and it is essentially a more cost effective method that minimizes the investment in resource preparation and storage by ordering and manufacturing the products right before the limit date required by the customer.

2.3.1.3 Flow Production

Flow or Mass production is the method that truly applies continuous assembly of goods. Individual products proceed sequentially from one to another manufacturing phase in a single production line; ach phase has specific staff and mechanisms that perform the same task over and over, as quickly as possible, with minimal quality loss.

Factories that apply this method are meant to be producing 24/7 with high product output so that the overall cost of running the production line is dispersed by each produced unit. Consequently, in cases of stoppage, there are always high costs associated, mostly due to resources needing to be reprocessed and disposed, in order to maintain quality levels in the start of the next work cycle [27].

As it is stated by its name, this method is used in large companies that have a very high demand but small cost flexibility per unit, therefore, being applied in areas such as energy, fuel, electronics, transportation, etc.

2.3.2 Structure and Types of DCS

A distributed control system is, by definition, composed by a hierarchical structure. For further explanation, we shall divide the general structure into three levels [24].

As it can be observed in Figure 2.17, the low-level is composed by the different industrial plant processes and their respective controllers. The processes can go from conveyors, robots, to ovens and are always equipped with easily reachable/compatible actuators and sensors. The controllers are usually custom microprocessors and are responsible for assuring that their respective processes successfully execute the planned manufacturing tasks.

The mid-level is where the supervision controllers are. Their task is to automatically oversee all of the plant controllers, set their work state and possibly detect global system failures.

Servers are installed in this level, which gather all the information available at low and mid-level and, consequently, send it to high level technologies. It is also the place where the lowest means of human intervention is possible by using equipment such as HMIs (screens with graphical interface of the process) and physical buttons.

The industrial network technologies and protocols used at this level are usually the ones supported by the installed HMI and controllers, for example,

serial type, fieldbus and industrial Ethernet communications (such topic will be described on the next sub-chapter 2.4).

The high-level is where specialized tasks are executed, having computers connected to an Ethernet LAN network with the goal of scheduling, coordinating and supervising the whole factory. Databases and remote clients are also available in this level, mainly for further accessibility and reliability.

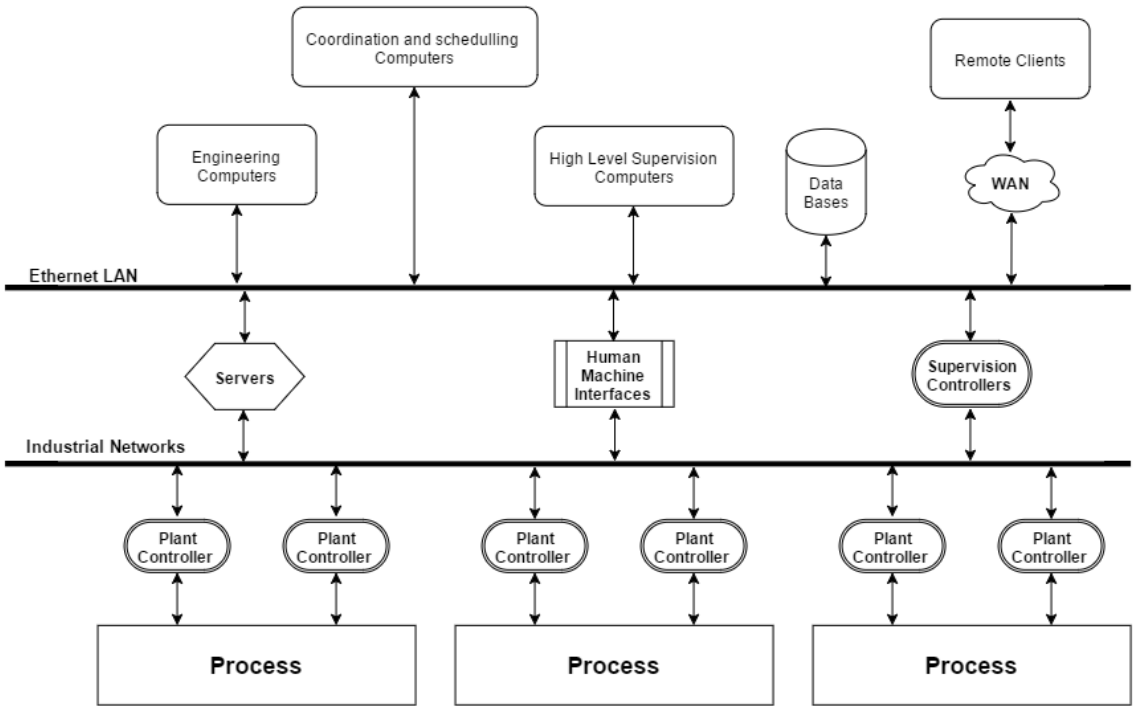


Figure 2.17 - Typical DCS structure.

On what concerns types of DCS, it is a very subjective matter but through research, it can be divided in the following categories:

- Discrete : this is where the conventional DCSs lie and they are basically conceptual systems that control processes of closed loop nature;
- Sequential: PLC based systems, typically illustrated by command control, where the notion of states and list of instructions is applied onto a process;
- Hybrid: applies both control concepts above;
- Smart: these are the most advanced distributed control systems that blur the conventional hierarchy structure by having a higher degree of interoperability and communication between all of its components, making each more independent and autonomous.

2.3.3 Comparison with other typical Control Systems

On the automated manufacturing scene it is important to compare and define the main used systems. Therefore, in this sub chapter, an overview and comparison will be made with SCADA and PLC systems.

2.3.3.1 Comparison with PLC systems

Comparing PLC to distributed control systems, they initially had different applications, PLCs were often used in simple batch controls.

PLC systems presented low component value where it was easy to reset the installation and possible downtime wouldn't present damage risk onto the process. In counterpart, the DCS was usually applied to much more expensive processes and would have to be able to control critical applications, where a possible failure has high profit loss and security risk associated [23].

In the modern days, it is actually hard to distinguish both systems due to the fact that the evolution of their technology and the adoption of new conceptual manufacturing methods has transitioned both technologies into each other [25]. Therefore being very usual to be observed the use of PLC as plant controllers in distributed systems.

2.3.3.2 Comparison with SCADA systems

A SCADA system is conceptualized for data acquisition with the purpose of plant supervision. It is mainly composed by software, being usually present in higher level layers within the hierarchy of an industrial system.

Its main focus is to gather information and present it to technicians in order for them to monitor and set the processes in correct work states, while DCS is a more process control oriented system.

Comparing both of the systems, as also seen in Table 2, besides DCS being process driven and the SCADA being event driven, there are other noticeable differences in typical systems. Distributed control systems are applied in smaller geographic areas, assuring reliable and good quality data and are composed by closed loop process control.

On the other hand, SCADA systems usually have larger geographical reach and the quality, reliability and determinism of the data is not high priority, since it is used only for monitoring purposes and not for autonomous control. It is also more efficient energetically, due to its equipment only being associated to data processing [25].

Another important factor is the human intervention at process levels. Distributed control systems allow operators to intervene in a more direct manner than a SCADA system would, which only allows HMI interaction. Moreover,, DCS are dependent of a constant stream of data from processes, in order to properly function, while SCADA systems are capable of keeping a well-functioning work state in case of outage, basing its decision on stored data.

Table 2 - Comparison between DCS and SCADA.

DCS	SCADA
Process driven	Event driven
Small geographic areas	Large geographic areas
Good data quality and reliability	Poor data quality and reliability
Powerful closed loop hardware	Power efficient hardware for data processing

Just like PLC and DCS systems identity is very blurry, nowadays, the same happens with SCADA and DCS systems. The common case is that SCADA components are already integrated on modern DCS, which present a very strong component of data gathering and processing.

2.3.4 Current State and Developments

One of the main trends in development on Distributed Control Systems is the conversion of its components into more intelligent interoperable units that focus primarily on the use of wireless communication technologies and networks, specifically in the input/output layer [28].

This layer is one of the most important parts of a DCS where countless process measurements and output actuation signals are read and applied in a single process. The current implementation of I/O devices (I/O Bus network) presents slow value transitions compared to the current developments in technology. The fact that it is based on a wired technology alone makes it more expensive in large industrial plants, while at the same demanding great complexity to assure reliable measurement/actuation of rotating equipment.

A second immerging trend is the implementation of server virtualization into the systems [28]. In other words, instead of having a centralized physical server from where every client requests information, that same server is further decomposed into several other virtual servers, which perform their individual tasks and allow access only to entities related to those same tasks. Benefits from this tendency come mainly from higher accessibility and less cost associated to physical resources.

Last but not least, there is also the idea of applying cloud computing into future DCS systems [28]. This would mean that, through remote servers, data would not only be gathered and transmitted, but also major physical tasks would be executed, completely taking away the notion of geographical barriers within a Distributed Control System.

As we can see, the major concerns in this subject relate to general security. In fact, relying on transmission of sensitive information through waves or relying on third party remote servers, if not well conceptualized, can be very risky. Valuable information to a company could be leaked or, even worse, manipulated. Thus, these trends are still far from being implemented on a large scale, especially considering that the automation industry is very conservative by nature.

2.4 Industrial Networking

The constant evolving of the industrial systems into more advanced digitally dependent structures, has also driven the need of developing new and better communication concepts that could fully adapt and elevate the overall performance of the systems.

Initially, the control structures only applied communications between the physical processes and the controllers. However, in the current times, networks are starting to be implemented at all levels, not only from a field control stand point, but also in a supervising and data gathering/processing point of view where the equipment focuses mainly in Ethernet standards.

2.4.1 Structure

Usually, the composition of a traditional industrial network has a significant number of layers and tends to be more complex the bigger the control system is (Figure 2.18).

In general, it is illustrated by a hierarchical structure composed by different communication protocols and media equipment where the lowest level of data transmission lies at the connection with the controllers and the physical world; above there is the application level that make the sharing of information between different protocols possible [29].

Following this, there is the supervision and monitoring level, ending on the data collection and external communication layer. This hierarchy has seen a tendency to dissipate, due to constant technological improvements, which facilitated the simultaneous interaction between all the components within a control systems [29].

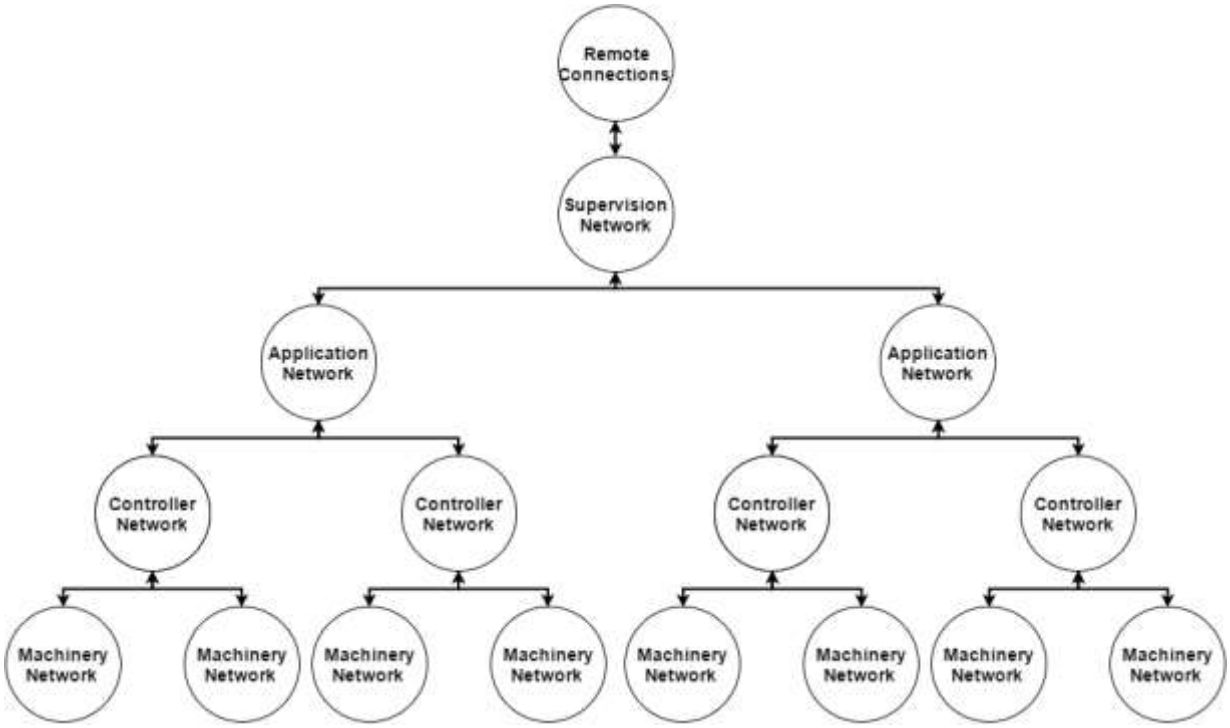


Figure 2.18 - A typical industrial network structure.

This structure, in order to be open and allow constant expansion must have a very well defined core topology. There are, essentially, three types considered [30]: Star, Bus and Ring configurations (Figure 2.19).

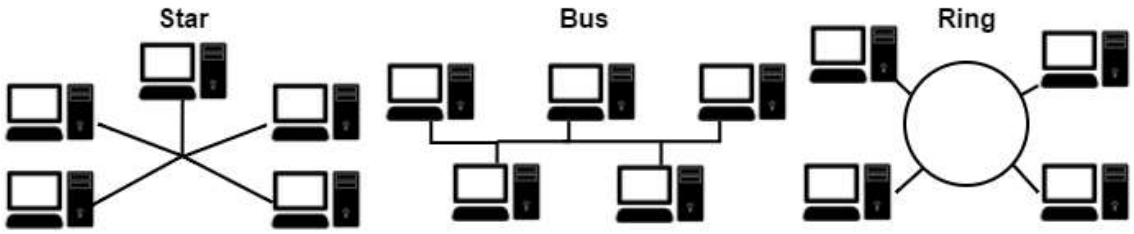


Figure 2.19 - Network Topologies.

The Star topology essentially illustrates a network that revolves around a central controller to which all the nodes/components are connected. This topology represents ease of component expansion or possible failure of a node without interrupting the well-functioning of the overall network.

Even though the failure of a node does not bring any consequences to the global system, this topology is highly dependent of the central controllers, therefore, their failure implies the failure of the entire network [30].

In the Bus concept, every component is connected to a common communication channel, hence, every message sent is received by all the nodes. A node failure does not jeopardize the global network, as long it is not damaging the bus.

The Ring configuration is essentially a cable that goes from node onto node, with the message sent and received between nodes until it reaches its destination. In this topology, a node failure shuts down the network.

On what concerns the equipment that is used to create the network itself, having a single cable is usually not enough. Because these networks must cover large plants, they need to be isolated and able to communicate with many different components with their individual communication specifications and protocols. Thus, an industrial network is usually physically illustrated by devices such as repeaters, servers, several routers and gateways devices [30].

The information transmitted within is usually decomposed in control, diagnostic, safety and historic type [29]. Control information refers to the data being traded between controller and physical instruments, usually inputs and outputs of a control loop. Diagnostic is the information collected from the different components of the system, in order to efficiently access and deduce how well the system is working and if there is any failure. Safety information is the one that is most prioritized. It is used to sustain critical tasks functioning and it has presented high reliability and real time requirements. Historical information is the general stored data that is usually used for prediction methods and historical analysis.

It is also important to point out that the industrial network's main emphasis is the ability to interact with the physical world, hence, the conceptualization of an operable network must have a critical focus on the following requirements [29]:

- **Safety and failure diagnosis**: since the network is used as a means of establishing communications between important control components and the physical processes, possible failure can translate into very serious consequences. Major losses of revenue can happen and, most importantly, the safety of the workers can be put at risk. Therefore, these networks must have very strong fault detection capabilities and very small packets of data to assure the quality of the information.
- **Real time response**: the speed associated with the data transmission must be fast enough to not only keep the system functionality maximized, but also to quickly re-send information in case of information loss. In general, the speed of transmission should be twice as fast as the receiving end requires it.
- **Determinism**: within a system there are several tasks that are time dependent, thus, being crucial that the network offers the capability of assuring low variance and predictable transmission times between

components, in order for a high degree of synergy and temporal consistency to be assured during events.

- **Periodic and aperiodic traffic**: there are many different tasks occurring within a system - some of them require a continuous flow of information in order to function (e.g.: control loops) and others only require it when an action is needed to be performed (e.g.: trigger events). Therefore, a network needs to establish different natures of transmission, depending on the information.
- **Ruggedness**: the network devices and wires must have a high protection index that assures the well-functioning of the network in locations where it is susceptible to adverse conditions such as dust, heat or vibrations.

2.4.2 Comparison with Commercial Networks

Comparing both networks, we can point out that, at their core, they have different main focuses: the industrial networks ultimate goal is to connect with physical machinery, in order to monitor and control a manufacturing task, while the commercial networks are only dedicated to processing and transferring data. Consequently, the networks have very different implementation requirements, as seen in Table 3.

Industrial networks have higher costs of implementation, demanding a varied array of dedicated and expensive hardware, while the commercial networks rely only on routers and cable. The structure of an industrial network has a higher quantity of layers associated, whereas the commercial network is only represented by the user local network that connects to a geographical site backbone and the latter connects to a supplier Wide Area Network [29].

On what concerns the requirements mentioned in the last sub chapter, the commercial networks do not need to offer a high grade of quality, they are intended to be deployed in corporate and domestic environments, therefore they offer low standards of ruggedness and failure severity.

Table 3 - Main differences between Industrial and Commercial networks.

	Industrial	Commercial
Main Focus	• Control of physical machinery.	• Transfer and process information.
Operating environment	• Manufacturing and distribution facilities; • Subjected to harsh conditions.	• Domestic and corporate use.

Structure	• Deep with highly hierarchized constitution; • Varied nature equipment and communication protocols.	• Shallow with uniform communication and standardized equipment.
Failure severity	High	Low
Reliability and Ruggedness	High	Moderate
Determinism	High	Low
Transmission times	250 µs – 10 ms	+50 ms
Temporal consistency	• Highly required.	• Not required.
Type of transmission	• Periodic and aperiodic transmission; • Small packages.	• Aperiodic transmission; • Large packages.

2.4.3 Protocol Overview

It is impossible to describe networks without mentioning communication protocols. Essentially, protocols are a set of rules that allow different devices to communicate and understand each other through a transmission medium.

Industrial network technologies and protocols can be gathered in three categories [30]:

- Fieldbus: it was the first means of communication, developed after the occurrence of the digital transition in the manufacturing industry. It is defined as a group of protocols that are mainly based on serial multi-drop data bus communication.
- Ethernet: highly penetrated technology in the business and domestic demographic, large data and high speed capabilities.
- Wireless: it is a tendency being developed and tested in current industrial environments, establishing a more adaptable and versatile control system.

Before heading onto a brief protocol overview, there is the need to go over a few important topics, starting by the main standardized protocol reference models (Figure 2.20).

The first developed model was called Open Systems Interconnection model (OSI), defined in 1984 by the International Organization for Standardization (ISO) [29].

This reference model is composed by seven layers:

- **Physical:** this layer concerns itself with the transmission of data over the hardware itself;
- **Data-link:** it organizes data and detects transmission errors between physical nodes;
- **Network:** manages data package routing;
- **Transport:** executes the data transmission between network nodes;
- **Session:** organizes and synchronizes the higher level data between network nodes;
- **Application:** is where the high level interfaces run on;
- **Presentation:** converts the information from the before mentioned layer to lower level layers.

The fieldbus network systems adapted the OSI model into the Enhanced Performance Architecture model (EPA) [29], consisting of only three layers: physical, data-link and application. This reduced model was conceptualized in order to eliminate non-essential delays within industrial networks.

It is also worth mentioning that following the Internet penetration in the market, the TCP/IP protocol was implemented and it represents a more efficient and simple OSI model for computer networking [29] - its constitution uses only physical, network, transport and application layers. It is still widely used as the foundation of the real time Ethernet existent nowadays.

Other important concepts often used are the client-server and producer-consumer models. Essentially, they represent how the communication is processed between the components of a network [30].

The client–server model (also called master-slave) illustrates a situation where a component controls the other in a unidirectional way; in other words, the client is the one who makes a request, and the server is the one who waits for the request and consequently responds to it. The most basic example of this communication model is a PLC controller directing a group of I/O devices.

The producer-consumer model is a broadcast model where a component sends a message (producer) and all the other components (consumers) receive it. If relevant to their tasks, the components process that information and act accordingly.

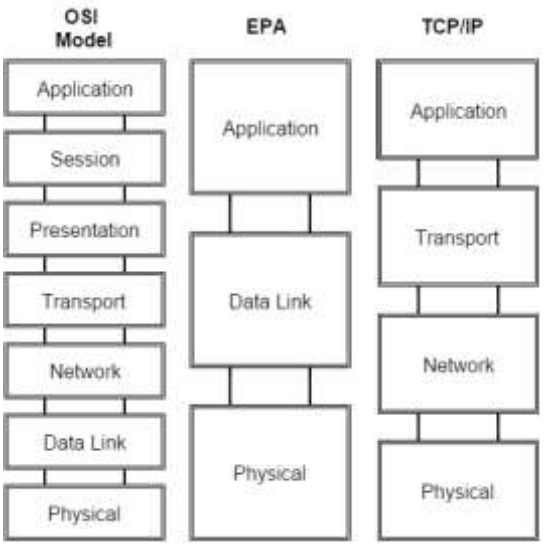


Figure 2.20 - Main protocol models.

Also worth noting is the way the data is transmitted while the communication is occurring (Figure 2.21). It can be:

- **Simplex:** the information is transmitted solely from the sender to the receiver;
- **Half-duplex:** both devices can send and receive information, but in a singular way;
- **Full-duplex:** that represents the half duplex concept but it is capable of doing it in a simultaneous fashion.

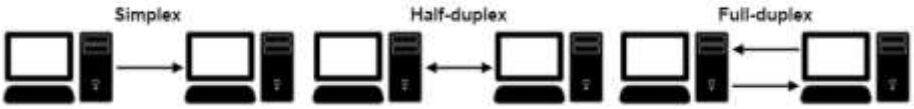


Figure 2.21 - Terminal connectivity classifications.

2.4.3.1 Modbus

The Modbus protocol was created by the company Modicon, in 1979, and its original goal was to allow point-to-point data transmission between PLC devices and their respective programing tools [31].

This protocol is classified as an application layer protocol and it comes in several different types, depending on the means of transportation used. The most used in the current day are the ones that run over serial and radio links, such as the Modbus ASCII and RTU, and the ones that run above Ethernet networks, like Modbus TCP and UDP.

Essentially, this protocol is based on the client-server configuration and in the Star network topology. It is constituted by only one client, typically a programming panel or a computer, and several servers which are controller devices mostly. The number of servers allowed on the protocol are limited by the number of serial ports the computer host has and by the limit number of 247 (maximum address number specified within the protocol) [31].

Following the client-server analogy, the way the Modbus protocol allows the transaction of information between devices is as follows: first, the client requests information; consequently, the destination server processes the request and responds accordingly. As an example: if there was a severe failure in one of the servers, the client would only be aware of that fact, when it proceeded to make a request to the server in failure, in the original protocol the server would not have permission to initiate a reply without a request.

The way the transaction happens between client and server can be observed on Figure 2.22, and for this transaction to occur, each server has to have a unique address identifying it, within the range of 1 to 247, the address number "0" is reserved as a global message identifier to whom every server has to reply to. On what concerns the client, there is no need for it to have an address, since there is usually only one [31].

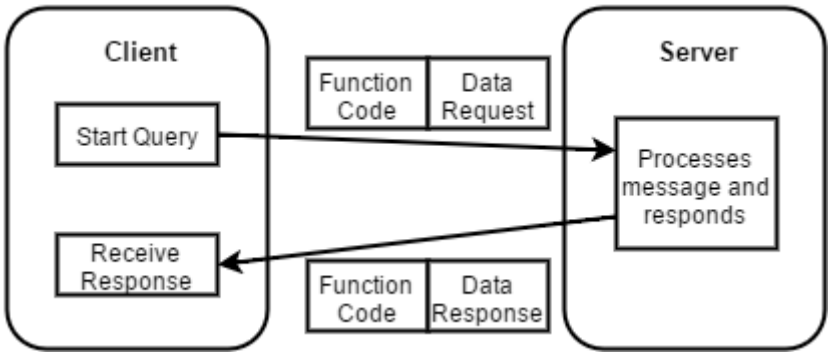


Figure 2.22 - Modbus protocol transaction.

Regarding the contents of the message, the message sent by the client is denominated by query and the response from the server is simply called response.

There are two modes of how a message is framed and timed in the Modbus protocol (Figure 2.23), the ASCII and RTU modes: the ASCII mode represents the data in such a way that a human can interpret it, while the RTU mode represents the information in a long chain of binary values. Besides timing the asynchronous frames, both modes have different overall sizes and error check methods.

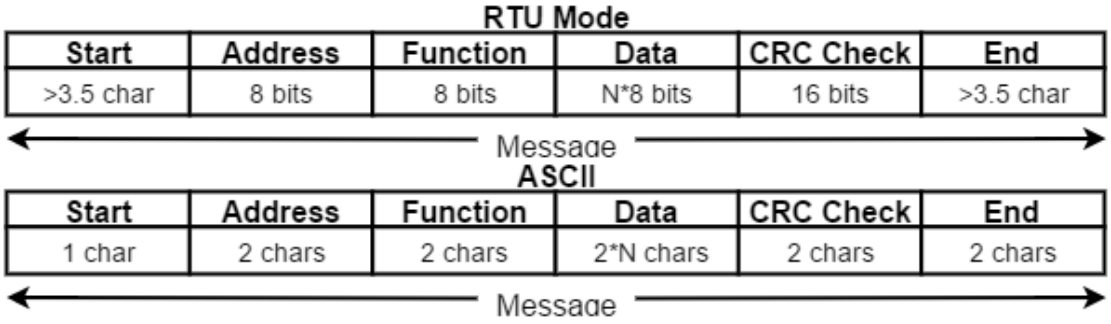


Figure 2.23 - Modbus message framing.

In both cases, the message is generally composed by the following structure [31]:

- **Device address**:
  This section of the message frame is intended to always hold the server identification and it is both used on queries and responses. Due to the message being sent in a broadcast kind of way, the address is crucial for the correct server to process it;

- **Function code**:
  The function code is a specific number that orders the server to apply an individual task or command. Depending on the function, the data portion of the message will vary.

  Concerning the functions themselves, they can be categorized in three main types: public, user defined and reserved. In Table 4 there are the most commonly used public functions offered by the Modbus protocol.

Table 4 - Modbus main function codes.

Function	Code
Read multiple coils	1
Read multiple discrete inputs	2
Read multiple holding registers	3
Read multiple input registers	4
Write single coil	5
Write single holding registers	6
Write multiple coils	15
Write multiple holding registers	16

- **Data bytes**:
  The data field will contain the necessary information to fulfill the initially function sent in the clients query. Depending on what that function wants

to manipulate within the destination server, the data bytes can be constituted by several types.

- **Error check**:
  This last sector of the message is used to deduce if all of the information was well secured and received. As mentioned before, there are different methods depending on frame transmission mode used.
  On the case of the ASCII, the error checking field contains two ASCII characters holding the result of the application of the longitudinal Redundancy Check Method (LRC). Regarding the RTU mode, the error check field is represented by two eight-bit bytes that serve as result of the application of the cyclical LRC.

2.4.3.2 Profibus

The Profibus protocol was one of the first created fieldbuses. It was developed by a consortium of various German automation companies in 1987, and due to the endorsement of Siemens it was widely implemented through all the manufacturing industry.

Its goal was to make possible the implementation of bit-serial fieldbus systems and it was defined has having three different profiles for specific applications. Firstly, Profibus-FMS for non-deterministic high level communications; secondly, Profibus PA (process automation) which is used for unsafe areas; and, lastly, Profibus DP (distributed periphery) for low level communication [33]. Since the main concept remains for all the profiles, this overview it will be focalized on the Profibus DP.

This protocol is based on a client-server bus topology where for every group of servers there are two different clients. The clients are classified in two classes [33]. Class 1 client is responsible for the central control tasks and the exchange of data with its servers and is generally represented physically by a PLC or a computer with special software. The class 2 client is usually a configuration software that is responsible for diagnosing and authorizing servers.

The information transaction starts via a periodical cycle where the class 1 client initiates a sequential request to every server on the network after the query-response interaction between the class 1 client and the servers, a token methodology enters in place, where the class 1 master passes the rights of interacting with the servers to the class 2 master. If a master doesn't have a token it is only allowed to read values from a server (Figure 2.24).

The way the message reaches the correct server, similarly to the Modbus protocol, is through identifying every server with an individual address in the

range of 1 to 125. Also, due to the fact that there are different masters within a Profibus network, masters do require address identification too.

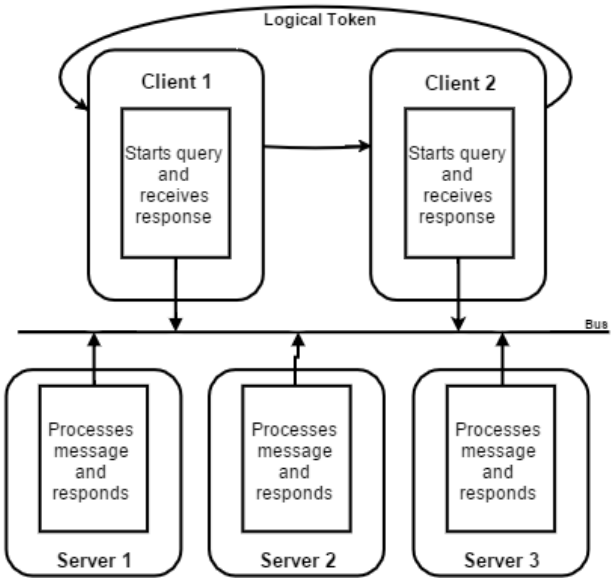


Figure 2.24 - Profibus protocol transaction.

The transmission applied uses two different types of services [33]:

- **Send and request data with acknowledge (SRD):**

  In this methodology, data is sent and received in one single transmission cycle. Strictly speaking, the client sends a request along output information and the server consecutively sends a response with input data. This service represents a very efficient way of exchanging data with I/O devices.
- **Send data with no acknowledge (SDN):**

  This service is used for global messages that need to simultaneously reach all of the servers (broadcast messages). They usually hold task instructions to whom the slaves act on but do not respond or give any acknowledgement to the client about who sent the message.

On what concerns the content of the message, there are two types of frames: request and response frames. The only difference between them, besides inverting who sends and who receives the message, is the fact that the request frame holds 1 byte with information about time synchronization between the master and slave involved on the transaction.

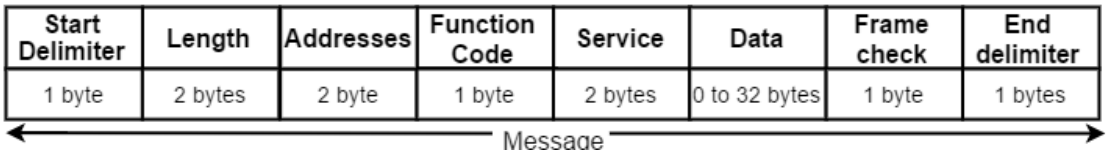


Figure 2.25 - Profibus message frame.

As seen in Figure 2.25, the typical structure of a frame is composed by the following elements [33]:

- **Start Delimiter (SD):**
  Establishes the start of the frame and the value it holds also defines the format of the frame.
- **Length field (LE and LEr):**
  It is composed by two bytes that present the size value of the frame sent.
- **Address sector (DA and SA):**
  This fields holds in 2 bytes, the addresses of both the sender, and receiver within the transaction occurring.
- **Function code (FC):**
  The function code is a specific number that orders the server to apply an individual task or command. Depending on the function, the data portion of the message will vary.
  Concerning the functions themselves, they can be categorized in two main groups: request message functions and response message functions.
- **Service field (DSAP and SSAP):**
  This field contains information about what type of transmission service is to be used by both a request message and response message.
- **Data field (DU):**
  The data field ranges from 0 to 32 bytes in length and it holds all the important information to be transmitted and processed between stations.
- **Error Check (FCS):**
  This byte holds a frame sequence that serves the purpose of checking if the message was well received. The error checking is based mainly in parity methodology: the sender calculates a parity bit for each consecutive eight bits composing the message; then, concatenates all of those parity bits into the error field. When the message is received, the server will perform the same action and compare it to the frame sequence within the message.
- **End delimiter:**
  Establishes the end of the frame.

2.4.3.3 *Profinet*

The Profinet protocol is a further development of the Profibus protocol onto the Ethernet. This adaptation was made aiming towards the elevation of the former into a higher degree of accessibility, compatibility and flexibility in industrial systems.

Its concept and structure are strongly similar to the Profibus overall concept. With that being said, it also uses a client-server bus topology with the same two different classes of clients assigned to a group of servers and that use token methodology.

One major difference is the fact that Profinet sustains a full-duplex means of communication in counterpart to Profibus that only sustains half-duplex. The other major difference is that within the transaction of information there is one additional communication channel in the Profinet.

From the two communication channels available in this protocol, there is the TCP/IP channel that is used for non-time critical tasks such as diagnosing and configuration. The other channel is used for real-time dependent tasks that need very fast and deterministic performances, such as main cyclic data processes, monitoring, alarming and critical task triggering.

2.4.3.4 *Open Protocol communications*

The Open Protocol Communication (OPC) was released in 1996. In essence, the OPC represented a highly compatible protocol that enabled universal connectivity and interoperability between all different manufacturer devices [35].

These devices often have their own unique data modeling structures, communication assessments and protocols, forcing the factories either to only constitute their whole production control systems with devices of the same brand or in order to take advantage of cost efficient measures forced to implement very clunky, complex and inefficient global systems full of manufacturer drivers for each different device.

This protocol concept can be viewed as a conversion layer between the communication of client and server devices (Figure 2.26). The devices do not need to be configured in anyway in order to exchange information - the OPC will simply pick the information of one device in a certain protocol and it will make it processable in the other device's protocol.

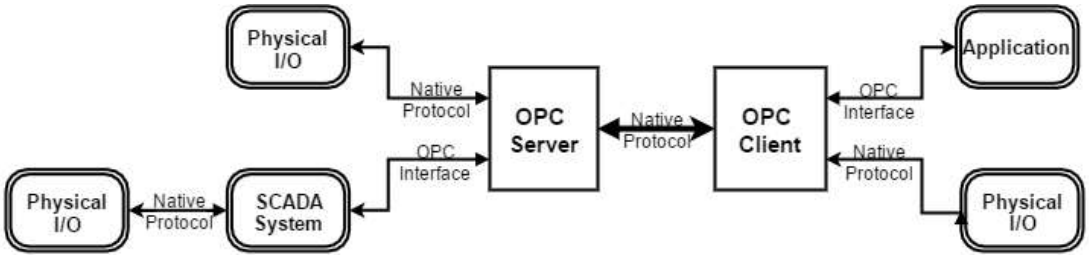


Figure 2.26 - OPC application scheme.

In order for that conversion to work, the OPC relies in a master-slave topology where both devices involved in the transaction are able to read and write onto each other, but it can only be started if the device that is being the client initiates such transaction.

Conceptually, the OPC server-client relation has the role of a translator between a data source and several applications connected to the OPC that require that data.

The OPC server usually is structured by the following components (Figure 2.27) [35]:

- **Open protocol communication**:

  This sector is responsible for assuring that the converted information is sent and well received by the OPC client
- **Native Communication:**

  This is where the OPC server communicates with the data source through its own protocol or custom made interfaces, gathering the data requested.
- **Translation module:**

  The translation module is the vital key of the OPC concept, since it has to efficiently be capable of interpreting a request from an application, convert it, send it to the native data source and, consequently, manage the data transaction between both different nature devices.

The OPC client has a similar structure but, instead of being connected with a native data source, is directly connected to a data application client.

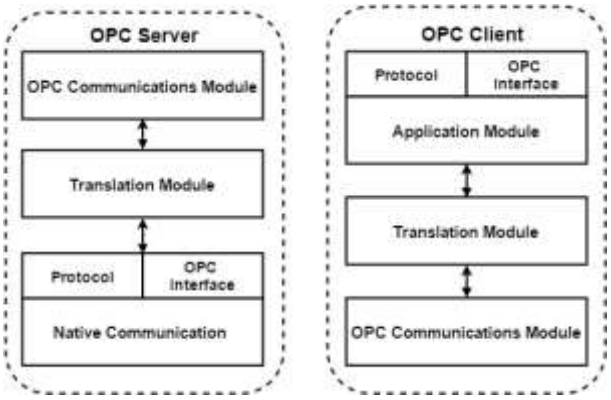


Figure 2.27 - OPC client-server conceptual structure.

2.4.4 Current State and Developments

Regarding current development efforts on what relates to industrial networks, there is the current trend of implementing protocols based in wireless technologies [29], with the general goal of furthering cost and complexity reduction in the quantity of wires in a global system.

Having a robust wireless network implemented into hazardous environments also brings numerous benefits in flexibility and installation longevity wise. The main reasons why it is yet to be largely implemented by the current industrial scene, lies on the lack of real time performance, determinism, limited distance reach and high susceptibility to interferences.

Information loss within an industrial factory can occur through numerous situations, from the transmission itself, where different sent signals interfere with each other, to industrial equipment creating electromagnetic charges and thermal noise, therefore making it very hard to design and implement a very efficient wireless network compared to a domestic/business application.

The most promising protocol standards that sustain the possibility of wireless networks being implemented in production factories are WPA-IA, Wireless HART, ISA100.11a and Wireless Mesh protocols.

The latter one allows for a dynamic network that can self-generate communication nodes, depending on the onset control system requirements.

2.5 Brief Chapter Conclusion

Regarding sub-chapter 2.1, the important factors to retain is that automation has been accompanying the human evolution trough the beginning of time, it is a concept that will only evolve, therefore, further developing production and manufacturing environments. Current researches revolve around the Industry

4.0 concept where communication, interoperability, autonomy and flexibility are key factors.

In sub-chapter 2.2, it is intended to pass informative knowledge on one of the most important devices applied in automation industry. It is described how they are structured and programmed. A very important fact, is that despite the technology's age, it is still very competitive device currently.

In Sub-chapter 2.3 it is important to note that distributed control systems are indeed the best control solution. Although more initial investment is needed, a lot of mid and long term benefits are gathered. Benefits like significant degrees of redundancy and information propagation, as also more open and flexible architectures. Developments in this area are associated to server virtualization, cloud computing and wireless technology.

Interoperability and interaction within a system is only possible trough highly networked systems, therefore in sub-chapter 2.4 it was decided to approach industrial networking and communication methods (protocols). The crucial information to retain from this sub-chapter is that industrial networks have much higher requirements comparatively to a network used for corporate and domestic purposes. Data reliability and speed is crucial for the well-functioning of DCSs.

The following project is strongly supported by all of the technologies described within this chapter. The application of industry 4.0 would simply not be possible without the combination and synergy between all of them.

3. Architecture and Implementation

In this chapter, a detailed description of this system will be issued, going over architectures, operational functionalities, specifications, to hardware and software implementation methodologies.

In sub-chapter 3.1, the global system architecture and functionality will be presented.

In sub-chapter 3.2, it will be descripted the conveyor control system in its entirety, from process specifications, operational functionality, to hardware and software implementation topics.

Following the same logic, in sub-chapter 3.3 the tank control system will be the focus.

Within sub-chapter 3.4, an overview over the supervision and high level monitoring component of this system will take place.

Sub-chapter 3.5, it will explain how the communication system was implemented and what tools it was based on.

Lastly, within sub-chapter 3.6, the remote clients implemented will be overviewed, explaining their functionalities and reason of their existence as well as how they were implemented.

3.1 Architecture and general functionality

This distributed system is strongly dependent on a local Ethernet network and software run in common computers, which facilitates the implementation of diverse client applications, therefore, allowing for a lesser dependability on specialized equipment.

This structure was mainly conceptualized to operate two physical processes: a tank and a conveyor. The reason for their use comes mainly from taking advantage of available resources in the laboratory and the fact that the

conjugation of those processes does sustain a realistic manufacturing contextualization. It is worth noting that each process is controlled by PLCs of different manufacturers, in order for the system to present a higher degree of interoperability (Figure 3.1).

Figure 3.1 - Hybrid System of Distributed Automation.

Besides the processes and the two controllers associated with them, the system is composed by a supervisor that monitors, imposes working states, gathers information and transmits it to remote clients. There is also a Human-Machine Interface device (HMI), which allows high level interaction with the whole system, having varied tools available for monitoring and global system manipulation. Furthermore, one Computer running an open protocol server, with the purpose of making the communication between controllers, supervisor and remote clients compatible and possible. And, finally, two remote clients that consist of one operational historian and an advanced monitoring client.

As seen in Figure 3.2, although the system may seem to be centralized, due to presenting a star architecture around the OPC server, on what concerns the control and system work states, every controller holds its own routines and procedures, making it distributed in a control sense. The main reason for it to be centralized on the OPC server, is due to the different manufacturer controllers, that have very different communication protocols, therefore making it imperative to have, in a communication sense, a centralized structure for them to be able to communicate with each other.

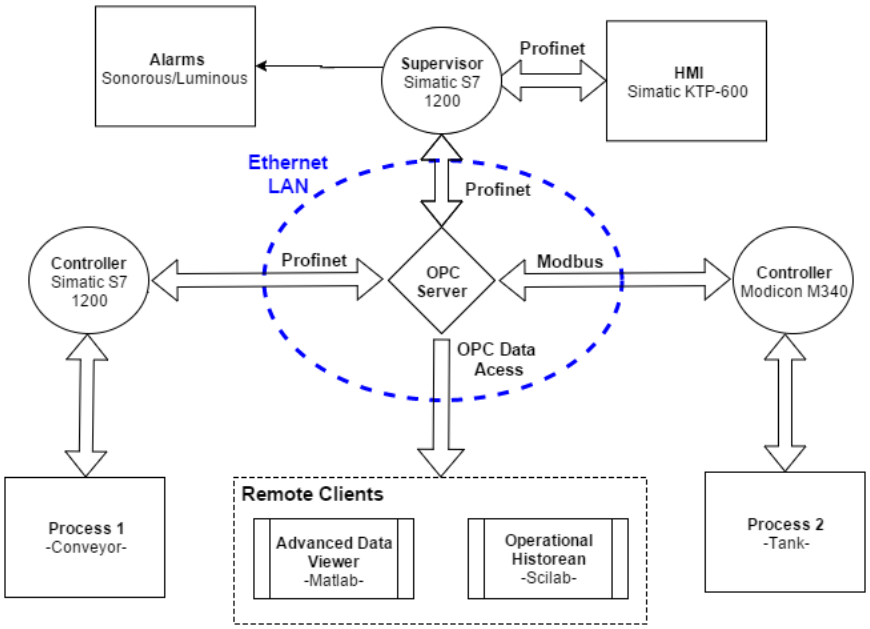


Figure 3.2 - Global system scheme.

Another important detail to mention is that the system does not hold a true unidirectional master-slave communication structure. All of the controllers have variables that are used exclusively to be received and others to be sent; with that being said, it can be seen as doubled master-slave communication between two controllers, where to a group of data, one plays the role of the slave and to other group of information the inverse case happens.

In regards to its general working, the system presents two main modes:

- Manual Mode: this mode permits the direct control of the physical actuators associated with each process and can be used either in low levels (via mechanical buttons near the tank/conveyor) or in high level (through the HMI).
- Automatic Mode: it represents the full automated cycle of transporting and treating a specific part, applying different working routines depending on the part's individual characteristics. Those part characteristics/specifications are pre-selected or created by a user before the initiation of the mode or during the treatment of a former part. This method can only be initiated if the whole system is operational. In case this condition is not verified, the whole system will maintain itself in a manual state. This mode has also a strong failure detection and treatment tools, alerting the whole installation in case of failure, waiting for a technician's presence in order to do manual monitoring and, ultimately, in case the failure is not attended, global system blocking is ensued.

3.2 Conveyor Control System

The conveyor implemented in this project has the main goal of transporting a part towards the tank station (Figure 3.3). Each part has its own individual characteristics and, since, some might be more fragile than others, this system was designed to allow the user to define the conveyor's speed in different stages of the transportation, assuring the most adequate transportation task to a certain type of upcoming part.

Figure 3.3 - Conveyor System.

3.2.1 Process Structure and specifications

Before describing the operational task implemented, it is important to mention which are the input/output mechanisms and part specifications available in the conveyor. As illustrated in Figure 3.4, there are a total of five sensors and one actuator surrounding a simple conveyor.

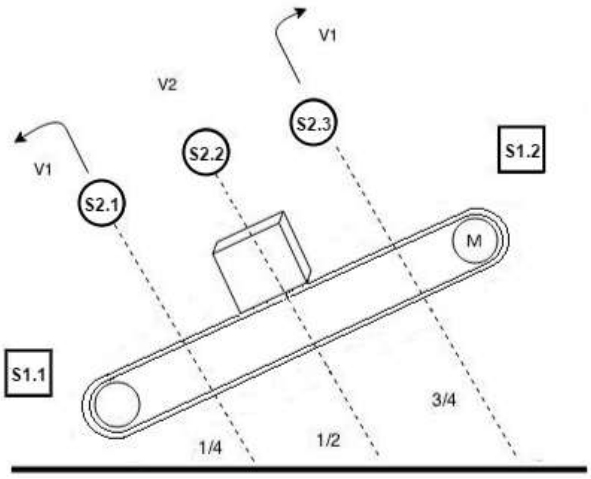


Figure 3.4 - Conveyor illustration.

The sensors are activation based, emitting a signal whenever the part carrier goes through the sensor's range of vision. The sensors on the edges of the conveyor (sensors 1.1 and 1.2) are mechanical limit relays, while the sensors 2 are optical. The information they transmit to the controller is used to know the relative position of the part, allowing self-calibration of the carrier and the application of user defined transportation speeds to the part in different sectors of the conveyor lane. The actuator is the motor that applies movement to the conveyor and allows the controller to manipulate the direction and speed.

Concerning the part specifications, the idea was to divide the conveyor path into two different sectors and associate to each different dedicated transportation speeds. In Figure 3.4 the two sectors can be observed, V1 illustrates a lower speed that assures the safety and well-functioning of the carrying task, at the pickup and deployment of the part. The sector from sensor 2.1 to 2.3 with V2 associated is meant to minimize the transportation time. The transportation speeds are limited in the range of 0.33 Hz/19.8 rpm or 0.1 m/s due to the small scale of the conveyor.

3.2.2 Operational Description

As observed in the Figure 3.5, the operational routine starts with the verification of several conditions before applying any mode of conveyor control. If the global system outside of the conveyor (specifically, the supervisor and the tank) is not active or responding, the manual mode is forced immediately. If there is any external emergency stop order or non-attended failure, the system will block itself.

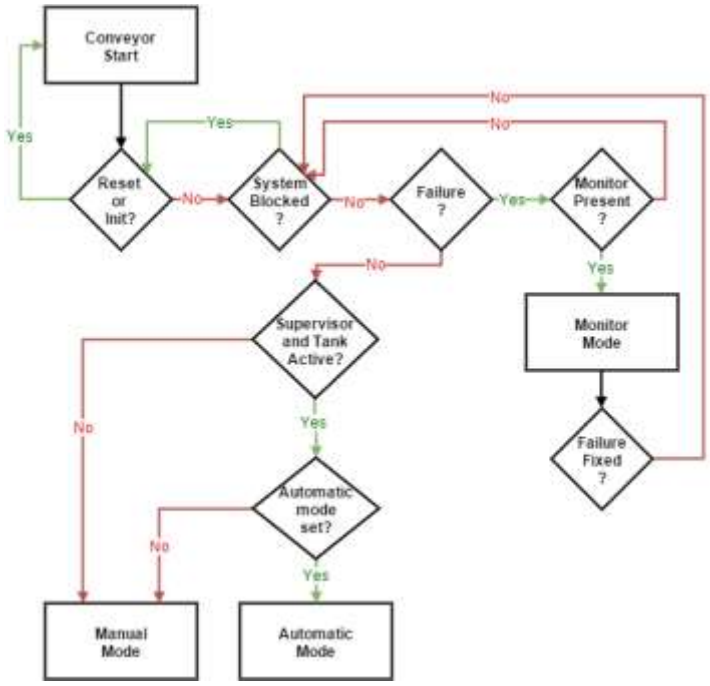


Figure 3.5 - Conveyor main sequential logic.

If there is a case of the conveyor being in a blocked state, the only way to re enable it is through ordering a reset via physical buttons or through external components (HMI). The reset executes a memory clear of all the important variables and applies a courier repositioning to the beginning of the conveyor belt.

When the manual mode is activated, direct control of the conveyor is made possible through either the low level interface or through the external HMI. This mode allows the technician to order the conveyor to stop or apply movement at a default speed going forward (from a previous workstation to the tank) or reverse. Additionally there are two other buttons that allow the increment or decrement of the speed. If the technician is monitoring the conveyor through the HMI device, there is also the option of applying a specific speed value onto the conveyor (Figure 3.6).

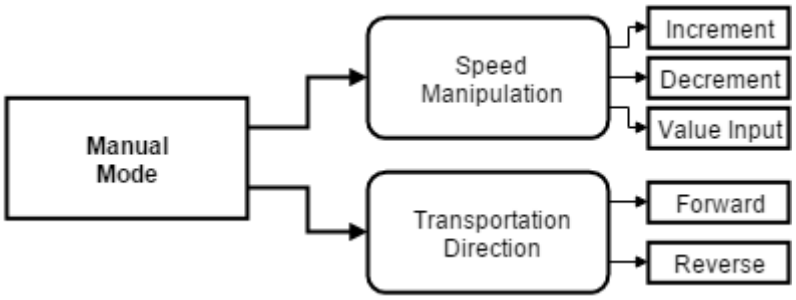


Figure 3.6 - Manual mode main features.

After assuring that the global system is all activated and successfully inter communicating, it is possible to activate the automatic mode through higher level interaction using the HMI. This mode starts by verifying if the tank is not busy and if there is a part ready to be transported. If those are valid, it will update the part specifications (V1 and V2) received from the supervisor and initiate the transportation process, as shown in Figure 3.7.

This transportation starts by checking if the sensor 1.1 is activated; in other words, if the carrier is located in the beginning of the conveyor, and, if it is not, it will start a routine to position the carrier on the initial position. Afterwards, it will wait a pre-defined number of seconds and execute the routine that takes the carrier to the end position (the tank location). Then, it waits again, takes the carrier to the initial position and starts the whole cycle again. The transportation routines always apply V1 before passing the first and last optical sensor.

One important last feature is the failure detection and consequent handling. In manual mode, there is no failure detection, since a technician is expected to be directly monitoring the conveyor (either through low or high level interaction), thus, being able to instantly detect and actuate upon the possible fault.

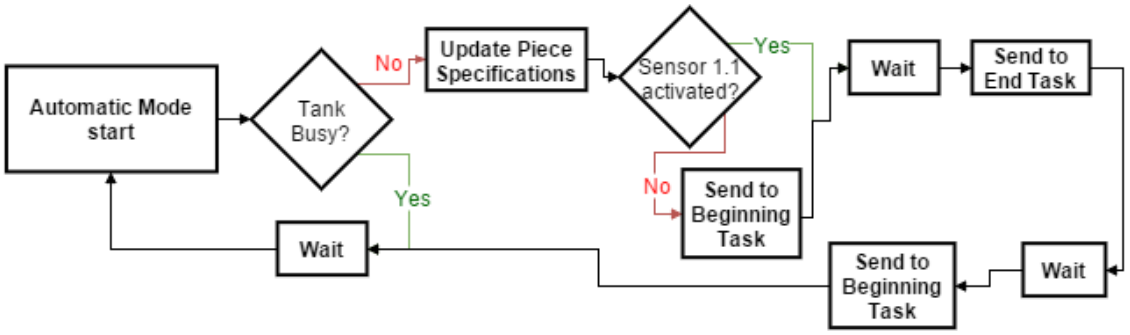


Figure 3.7 - Automatic mode logic scheme.

The autonomous failure detection is used within the automatic mode. In this project, there were two ways of checking a possible failure within the conveyor. The first one is through the application of a timer that starts counting every time a transportation routine engages; if the time limit defined is reached, the conveyor has been in motion for too long which means faulty sensors. The other method is through checking if the speed applied on the motor actuator is indeed the desired speed and, if not, it will point out to faulty inverters, motors or the conveyor structure.

If there is a failure detected, the conveyor process will send the signal to the server. Consequently, the supervisor will pick up that signal and proceed to alert all levels that the conveyor is faulty. It will stop momentarily the conveyor and wait for a technician to attend the failure (via low level interface near the conveyor or trough HMI). If the technician does not attend the failure, the global system will enter in a state of emergency stop. In the contrary case, if a technician is present, the system will go into a temporary manual state where the technician will test and deduce where the fault is, having three final choices: emergency stop the conveyor, shut down the whole system or ignore the failure, as seen in Figure 3.8. In case the failure is ignored, the system will continue its automatic mode routine.

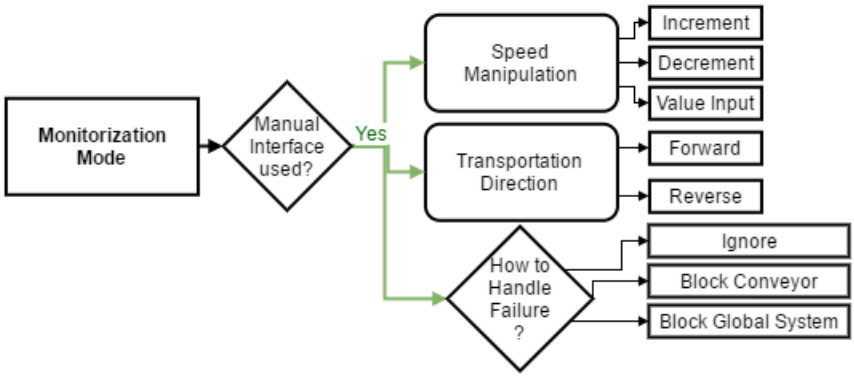


Figure 3.8 – Main features of monitoring mode.

3.2.3 Hardware Composition and Implementation

Figure 3.9 represents the real physical implementation of the conveyor process along the supervisor and HMI. The HMI and supervisor are meant to be in a different geographical locations from the conveyor process but, for practical purposes, it was decided to mount both in the same panel.

On what concerns the controller, a Siemens s7-1200 was used. This PLC features two racks of eight digital Relay inputs of 24 V DC and two other for digital outputs of 24V DC (total max current output of 2A), one analog output that ranges voltages from 0 to 10 V DC and, finally, one standard Ethernet port.

The way the controller actuates on the conveyor is through indirectly manipulating an inverter, which consequently direct controls the motor. The controller sends a signal to the inverter that goes from 0 to 10 V through the analog output. The inverter is previously configured to interpret that signal in the following way: if the value received is in the range of [0 to 5[V it will apply the reverse direction, being 0V the top reverse speed; if the value is] 5 to 10] V it will apply the forward direction onto the conveyor, being 10V the top forward speed. The 5 V value represents that the motor is stopped.

In order for the inverter to offer both reverse and forward movement in a single range of voltages, it creates the issue of having to apply a specific voltage in the middle of that range for the motor to be stopped, which brings potential malfunctioning in cases where the inverter is turned on and the controller is not, due to the fact that the inverter will receive 0 V signal from the controller and interpret it as the right signal to apply maximum reverse speed onto the motor.

The inverter used was a delta VFD-L. It is a very simple inverter that can apply a top speed of 400Hz or 24000rpm, and it can either control DC motors or three phased AC motors. The justification for its use comes from two main reasons: motors mostly require high power drives and the motor used is a three phased AC motor, both of which functionalities the PLC does not and is not meant to provide.

One very important matter about the inverter, is that it was configured as controlling a range of speed around 40Hz or 2400 rpm, but the motor has a reductor box coupled that converts those 40 Hz into a much slower speed of 0.33 Hz, 19.8 rpm or 0.1 m/s.

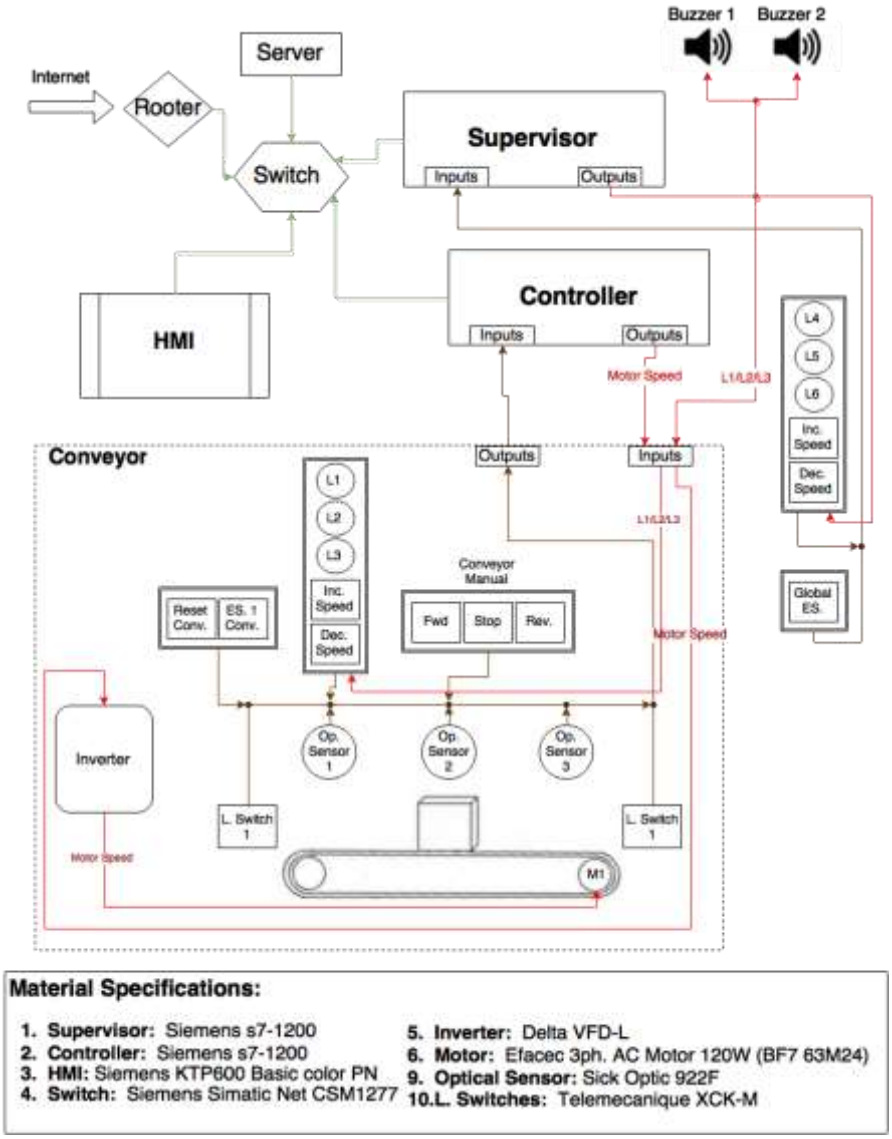


Figure 3.9 - Hardware configuration scheme.

Concerning the motor itself, it is a 120W Efacec three-phased AC motor, picked mostly as it was the available resource, therefore presenting specifications that weren't really demanded, such as a reductor box and significantly more power than needed for the small conveyor process.

Regarding the sensors used and, as seen in the Figure 3.10, the two in the edges of the conveyor belt are the Telemecanique XCK-M, which are simple contact limit relays, and the others are Sick Optic 922F sensors, which are photo-electric based. Both sensors are used in this project to work with 24V DC.

The optical sensors are composed by an infra-red transmitter and a photo electric receiver. Taking advantage of both, the way they were used as a position detection sensor was through attaching a reflector material onto the conveyor carrier, making the sensor identify its own transmitted photo-electric beam. In

the Appendix sub-sections A and B, there is the complete connection scheme of the panel and I/O tables.

Figure 3.10 - Sensors used within the Conveyor (contact relay and infra-red optical sensor).

3.2.4 Software Methodologies

For the programing of the controller it was used the proprietary's software Step 7® that is offered within a global tool software named Siemens TIA portal. As it can be seen in Figure 3.11, the software is mainly composed of a script sector where the executable tasks and functions are implemented, and a sector for variable definition and debugging.

As in most of the PLC software environments, the variables are all pre-defined in tables where it is chosen the type of data used and the specific memory slot. Concerning the code implemented, Step 7 allows the three main languages from the IEC 61131-3 standard: Ladder, Function Blocks and Structured text.

Structured text language was chosen exclusively in the conveyor programming, due to the fact of many concurrent sequential actions being used, which, ultimately, would provoke a very high complexity degree by using the other two graphical languages.

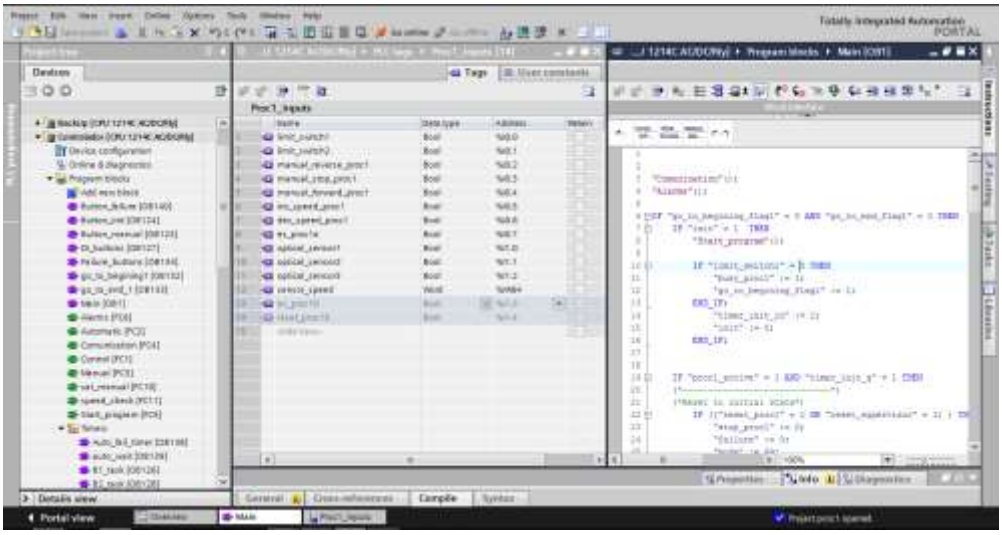


Figure 3.11 - Step 7®/ TIA Portal® Software environment.

The code was essentially divided into four task groups:

- Communication: it has the function of updating the memories that are used to be read or written by the server. This task also processes the values in order for them to be interpreted correctly. The processing relies mainly on normalizing the value range and converting its representation type.
- Low level interface management: its function is to manage both physical button inputs and light outputs to give a simple and effective low control and interpretation of the conveyor in its nearby confines.
- Conveyor control: this task is where the major bulk of executable actions are applied. Its job is to manage the manual and automatic mode of the conveyor, but also to monitor and provide tools to identify, alert and correct possible failures.
- Transportation Routines: dedicated trigger tasks for autonomous transportation.

The conveyor transportation procedures (take carrier to tank and take carrier to the beginning) were made into concurrent trigger tasks, so the controller would not see itself stuck performing that sequential action only, and completely ignore all the other important functionalities. As seen in Figure 3.12 , the task will only start if the "go to beginning" flag is set. When the beginning sensor or limit switch 1 is activated the flag will be reset, meaning that the destination was achieved and the task will be forced to stop.

```
IF "go_to_begining_flag1" = 1 THEN
        "timer_transport_in" := 1;

        IF "limit_switch1" =  1 THEN
          "go_to_begining_flag1" := 0;
```

```
            END_IF;
            IF "go_to_begining_flag2" = 0 THEN
                IF "mode" <> 1 THEN//default speed values for manual mode in
                                        //the range of [0 to 100}%;
                  "v1_reverse_proc1" := 77;  //speed in %
                  "v2_reverse_proc1" := 77;
                END_IF;
              "setpoint_speed" := "v2_reverse_proc1";
              "go_to_begining_flag2" := 1;
            END_IF;

            IF "stop_proc1" = 0 AND "es_supervisor" = 0 AND "go_to_begining_flag1"=1 AND
"failure" =0 THEN

              // Adjusts the motor speed acordingly to the position it is
              IF "optical_sensor3" = 1 THEN
                "setpoint_speed" := "v1_reverse_proc1";

              END_IF;
              IF "optical_sensor2" = 1 THEN
                "setpoint_speed" := "v1_reverse_proc1";

              END_IF;
              IF "optical_sensor1" = 1 THEN
                "setpoint_speed" := "v2_reverse_proc1";

              END_IF;
              "q_motor" := REAL_TO_INT(16800 - 11200.0 * INT_TO_REAL("setpoint_speed") /
100.0);
              END_IF;
ELSE
   "go_to_begining_flag2" := 0;
END_IF;
```

Figure 3.12 - Partial code of a transportation task.

The conveyor control was completely designed to follow a sequential command logic, therefore, there was no prioritized focus in closed loop control of the motor speed or carefully studied and forced sample times. The sample time used is the one pre-defined by the PLC which, according to the s7-1200 PLC manual, is a conjecture of the cycle time and I/O modules update rate. After checking the I/O module's properties is estimated to be 12.8 and 40 milliseconds, for digital and analog connections, respectively.

In regards to the speed applied onto the conveyor and as seen in Figure 3.12, the speed has to go through a lot of conversions until it is ultimately put into action. From a software stand of point, the goal is to send into the inverter a signal that scales from 0 to 10 Volt, which corresponds in a processed value of 5600 to 27648.

From 5600 to 16800 corresponds a reverse direction of the motor, being 5600 the conveyor top reverse speed of 19.8 rpm, the value 16800 corresponds 0 rpm.

From 16800 to 27648 corresponds a forward direction, being 27648 the top forward speed.

Within the program, for simplicity and abstraction purposes, both reverse and forward ranges were converted into the range of 0 to 100% and since both ranges are not equal, two distinct linear conversion equations were applied.

3.3 Tank Control System

The tank used in this project has the main focus of applying a timed submersion of a welded part where there is feedback control applied into maintaining a specified water level (Figure 3.13). Depending on the degree or risk of damage liability associated to a specific welding technique, the tank controller was designed so it would allow the selection of several types of control methods upon the liquid submersion, treatment limit times and the level of the tank desired to be filled. As observed in Figure 3.13, the tank also presents low level interaction nearby.

Figure 3.13 - Tank system.

3.3.1 Process Structure and specifications

This tank, although very simplified, does not hold a linear behavior through different ranges of water level, being more unstable especially at the lower and upper extremities of the tank.

As Figure 3.14 shows, the tank process is mainly composed by one sensor and four actuators.

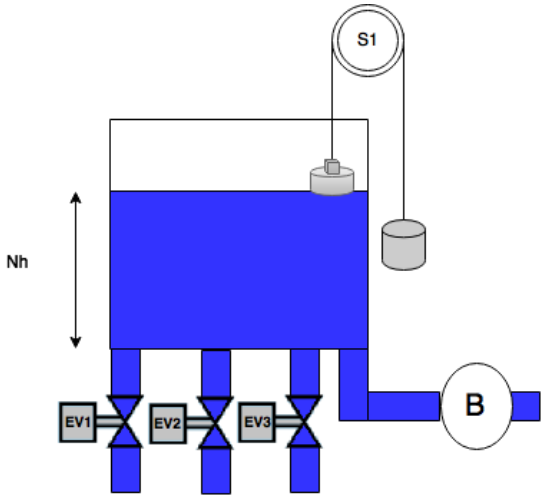


Figure 3.14 - Tank illustration.

The sensor is used to deduce the water level within the tank. It is composed by a wire attached to a floater and a weight, a sheave and a potentiometer directly concatenated to the former. The weight will keep the wire always extended while the float accompanies the water level variation. The sheave will rotate accordingly to the wire response to a water level variation and, as consequence, the potentiometer will generate a specific DC voltage.

The actuators are a water pump and three electro valves. The water pump is the only means of supplying the tank with water and is directly associated to a DC motor that allows the controller to manipulate the supply. In this project, however, its range was limited between 0 to $10.6 \times 10^{-5}\ m^3/s$ (8V motor supply)

The electro valves have the function of limiting the speed with which the water escapes the tank, being each associated to one of the three escape tubes of the tank. The tubes associated with valve 1 and 2 have the same size and are significantly larger than the tube associated to valve 3.

Regarding the part specifications, the idea was to let the user select important factors associated to the pickling treatment, therefore the time limit, desired water level and type of controller were the personalized characteristics chosen.

Five different controllers were implemented: command control, which is the only no water renewal sequential type control, and four others that relate to different variations of feedback control using mainly PID methodologies, as described later on.

3.3.2 Operational Description

Similarly to the conveyor programmed logic, and as seen in Figure 3.15, the main operational routine always begins with variable resets and state variable

checks; more specifically, it verifies if the other two main components are communicating (supervisor and conveyor) or if there isn't any stop state or failure being issued. If by any means the system encounters itself in a blocked state, only a reset will restart the process.

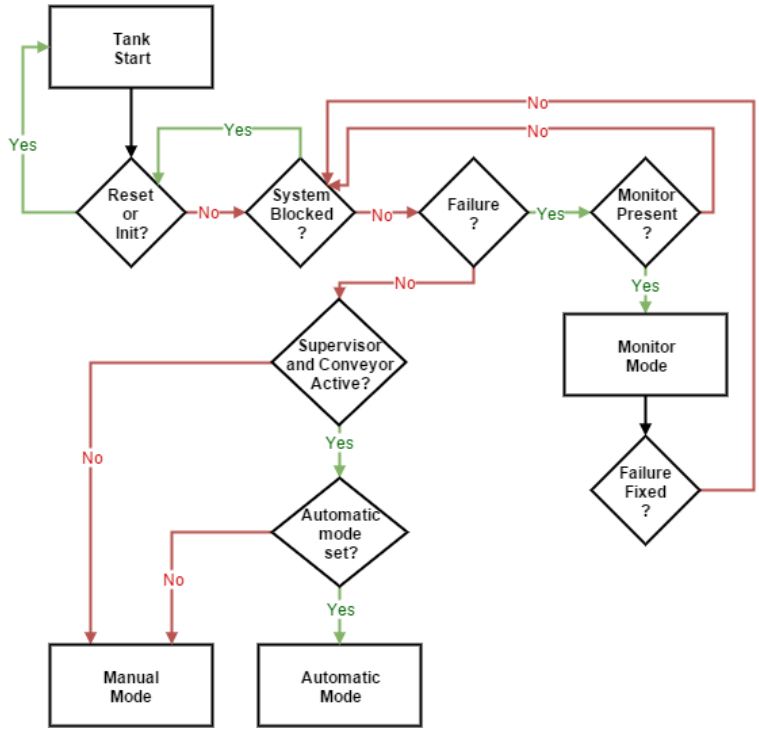


Figure 3.15 - Tank main logic scheme.

In the case of manual mode being activated, the tank can be directly actuated by the user through the use of a switch panel nearby the process or through the HMI. The functionalities available are the opening or closing of each electro valve and the activation and the speed of the water pump (Figure 3.16).

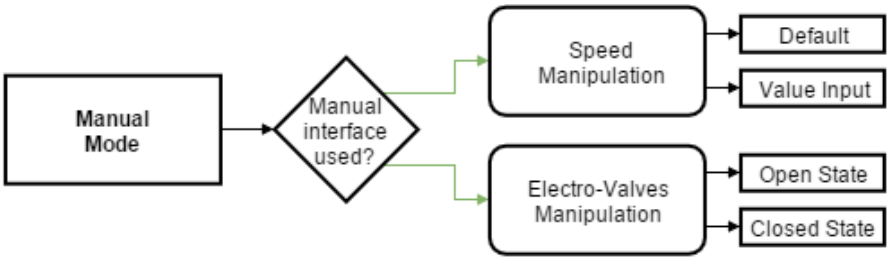


Figure 3.16 - Manual mode main features.

Having the whole system operational and inter communicating, the activation of the automatic mode is possible (Figure 3.17). This mode starts by verifying if any part has arrived to the tank station, which is done through receiving information from the conveyor about its end limit sensor state. If the former condition is verified the mode will update the parts' characteristics specified by the HMI and will initiate the treatment routine with those in mind.

After the treatment time being achieved, the automatic mode will start over again.

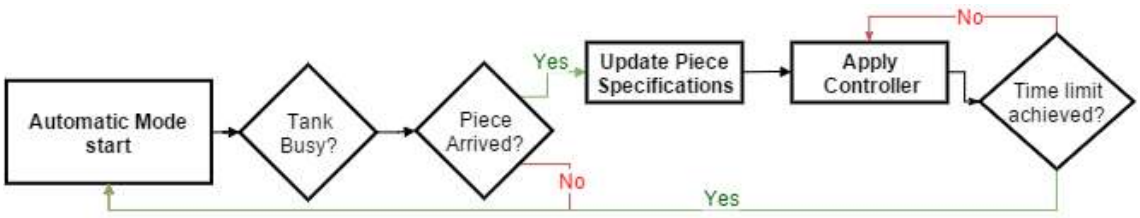


Figure 3.17 - Automatic mode logic scheme.

The treatment routine execution will vary depending on the water level controller applied and as mentioned before the selection of the following controllers are possible:

- **Command:**
  It uses a simple sequential logic where the tank remains static after achieving the water level specified.
  The way it works is as follows: starts by closing all electro valves and activating the water pump at a default value of $5.3 \times 10^{-5}\ m^3/s$ (4 V DC supply); when the tank reaches the specified value it will stop, in case the tank presents a level of water higher than 2%, compared to the desired a level of water higher than 2%, after the pump is stopped, the valves will open and close until the level is rectified.
  It was conceptualized for using in situations where the user does not see the need to have liquid renewal for each part.

- **Relay:**
  Uses a simple on-off methodology. If the water tank level is below the desired; the water pump is activated to maximum rotation symbolizing $\mathbf{10.6 \times 10^{-5}\ m^3/s}$, otherwise, the pump will be off and the water will escape through the tube associated to valve 1.
  This controller was conceptualized for users that have robust actuators and value a less precise/efficient type of controller and prioritize faster execution.

- **PI controller:**
  As seen in the Figure 3.18 and equation (3.1), this controller is based on a closed loop analogy that starts by calculating the error between the tank level read from the sensor and the desired tank level (y(k) and r(k)). The error (e(k)) will be multiplied by the sum of the proportional and integral components of the controller. The result of that calculation will generate a corrected value to actuate onto the motor.

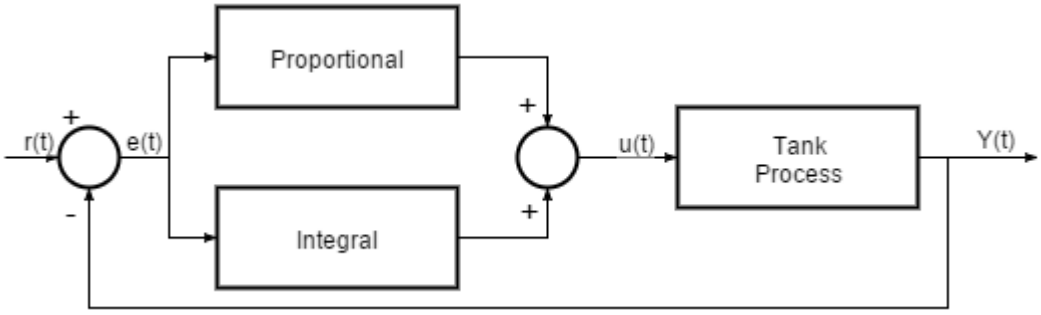


Figure 3.18 - PI controller block scheme.

The proportional component of the controller is represented by (3.2) and its focus is to influence the current error value. On the other side, the integral component of the controller is represented by (3.3) and its goal is to process the error while taking into account the error samples of the past.

$$\boldsymbol{u(t) = P(t) + I(t)} \tag{3.1}$$

$$\boldsymbol{P(t) = Kp * e(t)} \tag{3.2}$$

$$\boldsymbol{I(t) = Ki * \int_0^t e(\tau)\, d\tau} \tag{3.3}$$

This controller needs to be tuned previously to its application, by calculating and simulation testing of the correct values for the controller gains Kp and Ki (proportional and integral gain).
Firstly, in this project, the Ziegler-Nichols method was used to provide a notion of the values required; the next step was to manually adjust those values by testing the controller on a simulated model.
The values chosen for this controller were Kp=2.5 and Ki=0.156.

- **Switched PI controller:**
  The difference that stands between this controller and the former is that instead of using the same controller gains for the whole tank, there were different parameters distributed into smaller tank level intervals, with the intent of maximizing performance and stability (Figure 3.19).

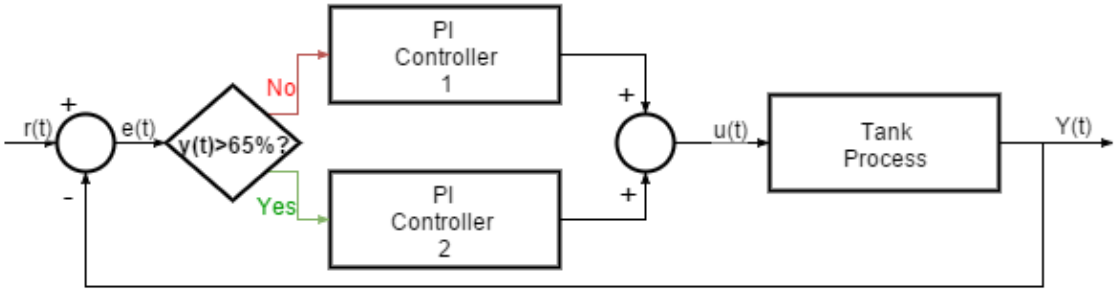


Figure 3.19 - Switched PI controller block scheme.

In the modeling stage, it was decided to divide the tank level water control into two ranges: [0 to 65]% with (Kp=4, Ki=0.27) and]65 to 100]% with (Kp=4.5, Ki=0.3).

- **PI controller with model rectification:**
  This controller adds an internal model of the process to the first PI controller, having the goal to further help rectify the set-point error. The idea is to change the initial specified set-point (r1(t)) with the difference between the predicted water level and the one read from the process (Figure 3.20).
  The model used was the same used for the simulation, and it is, essentially, a trained neural network based in black box modulation.

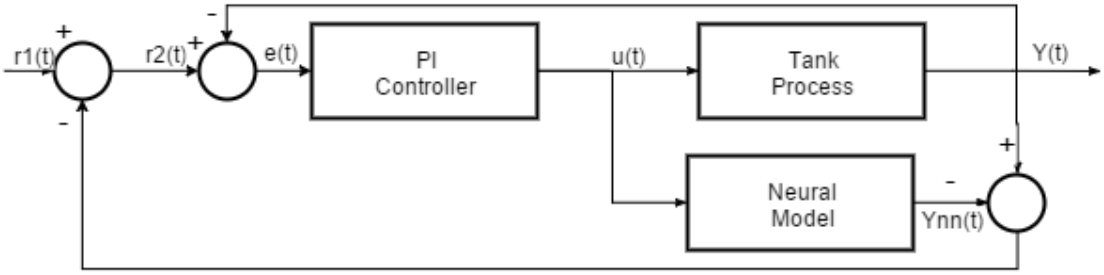


Figure 3.20 - Predictive PI controller block scheme.

It is worth noting that the derivative effect was not used in the above controllers, due to the fact that satisfactory results were achieved during modeling phase, with only the use of proportional and integral components of the PID.

On a last note about the automatic mode, is that the closed loop controllers, in the beginning of their routine, always open the electro-valve 1 and close the rest.

Concerning the failure detection, it holds a similar analogy to the conveyor control system: in manual mode there is none, since it is conceptualized to have a technician directly monitoring. Within the automatic mode, a failure is detected mainly through the incapability of the controller applied to get the water level up to the desired set-point, after the dynamic state has ended. This failure points out multiple possible failures such as water pump, sensor, electro-valves malfunctioning or simply hardware degradation that influences the nominal behavior of the process. If the failure is indeed detected, the system will halt itself, inform the supervisor, and set the alarm on its own process confines. If a technician is present, the system will head onto a variation of the manual mode, called monitoring mode (Figure 3.21). If not, the system will head onto a global system emergency stop.

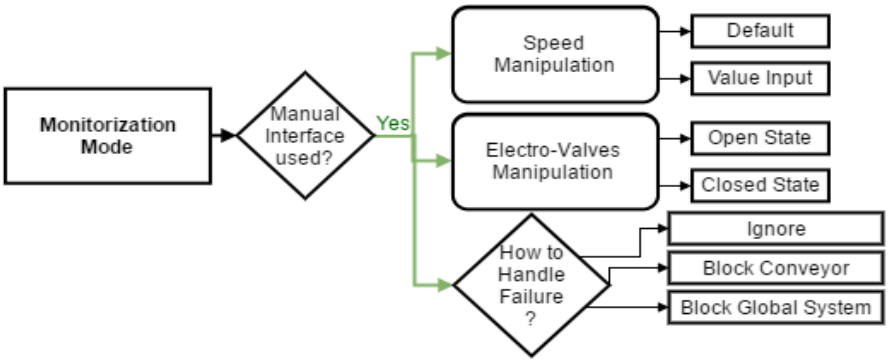


Figure 3.21 - Monitoring mode main features.

3.3.3 Hardware Composition and Implementation

Figure 3.22 illustrates the main tank control system, the controller used is a Schneider M340 PLC equipped with one analogic, two digital I/O modules and Ethernet, Modbus and Can-Open ports.

The analogic module is an AMM0600 that holds four inputs and two and outputs with a range of [0 to 10] V DC. The digital modules are DDM 16022 that present eight inputs and eight outputs of 24 V DC.

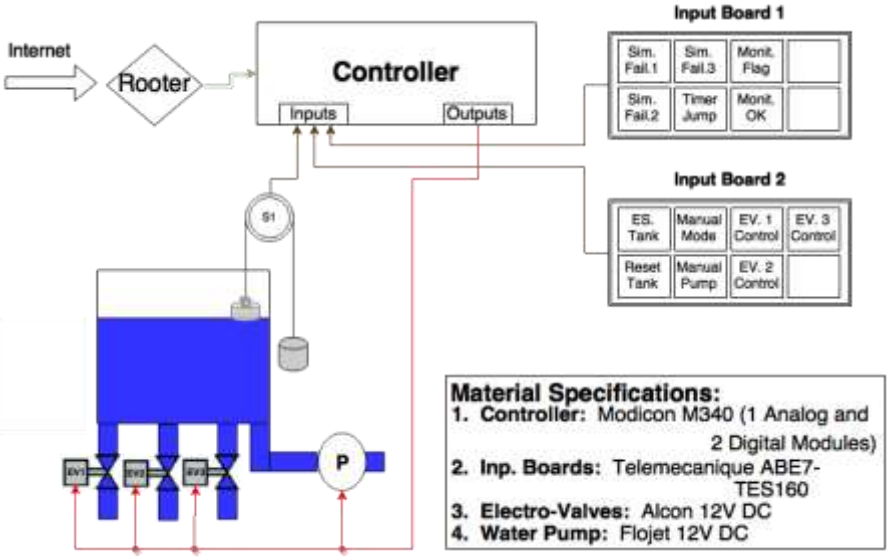


Figure 3.22 - Tank hardware configuration scheme.

Concerning the actuators, the water pump is a Flojet 12V DC. In the present project, the signal transmitted to it by the controller was limited to [0 to 8 V] or $[0\ to\ 10.6 \times 10^{-5}]\ m^3/s$. This limitation was implemented so it would prevent unnecessary harm and noise caused by the controller methodologies applied. Moreover, before the signal reaches the pump, it goes through a motor driver to assure enough power reaches the actuator and protect the PLC.

Due to the fact that the digital outputs coming from the controller are 24V DC, there is an electronic circuit that scales the voltage down to 12V.

In regards to the water level sensor, it generates an array of DC voltages depending on the potentiometer position and presents a non-linear representation of the tank water level having the [3.83 to 9.32] Volt range translated onto [0 to 100]%. In the Appendix sub-sections A and B there is the complete connection scheme of the tank panel and I/O tables.

3.3.4 Software Methodologies

For the programming of the M340 controller it was used the Schneider proprietor software Unity Pro®. As it can be seen in Figure 3.23, it holds a similar presentation and structure comparatively to the Siemens programming software Step 7®. What distinguishes the M340 programming software is the lack of graphical human-machine interface programming and holds the complete IEC 61131-3 programming language library; in other words, it is possible to program the Controller in Instruction List, Sequential Control, Structured Text, Ladder Logic and Function block, unlike Step7® where you can only use the latter three.

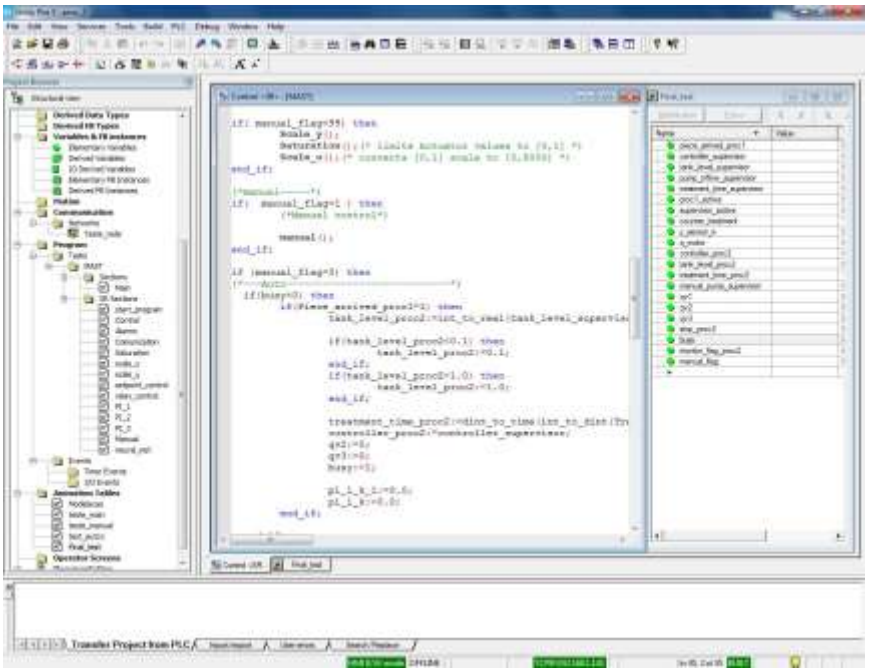


Figure 3.23 - Unity Pro® Software environment.

One important factor to note is that, since the main task is running itself as a cycle, there are no functions within it that hold typical cyclical instructions such as “while” and “for”. Therefore, an instruction that needs to function during a significant period of time will be called several times by the main task, instead of the case where the main task locks itself on that function.

Having the factor above taken into account and not repeating too much of what was said from the conveyor software methodologies sub-chapter 3.2.4, the

code implemented within the main task divides itself in: communication, low level interface management, failure detection and ultimately the tank control.

On what concerns the execution sample time, it was forced a sample time of 200 milliseconds due to the fact of the tank not having a fast dynamic behavior.

Regarding the automatic controllers implemented, The PI code implementation was heavily based on K.J. Aström and T. Hägglund discrete logic [36].

As seen in Figure 3.24, the PI controller was indeed protected against the eventual windup effect. This effect is illustrated by the continuous increment of the integral component, which leads to the eventual malfunctioning of the controller. There are many different ways of tackling this problem, being the most common one the periodic reset of the integral component. However, in this project, it was decided to simply saturate the integral component within an interval where the controller performs accordingly to the conceptualized. This approach was used so the controller would not show the periodical set-point error associated with the resetting of the integral component.

```
pi1{
        Qv1:=1;

          pi_i_k_1:=pi_i_k;  (*update for the integral component*)

          pi_bi:= (pi1_kp*(time_to_real(Ts)/1000.0))/ pi1_ti;  (*integral gain*)
          pi_ao:=  (time_to_real(Ts)/1000.0)/pi1_tt;     (*gain associated to actuator saturation*)
          pi_p_k:=error_k*pi1_kp; (*proportional component update*)



          u_motor_k_p1:=pi_p_k+pi_i_k_1; (*calculation of the controller actuation*)
          u_sat_pi:=u_motor_k_p1;

          if(u_sat_pi>1.0) then  (*actuator saturation*)
                    u_sat_pi:=1.0;
          end_if;
          if(u_sat_pi<0.0) then
                    u_sat_pi:=0.0;
          end_if;
            (*calculation of the integral   component for the next sample*)
          pi_i_k:= pi_i_k_1+pi_bi*error_k+pi_ao*(u_sat_pi-u_motor_k_p1);

        (*Saturation of  the integral component to prevent the wind up effect*)
          if (pi_i_k>5.0) then
                    pi_i_k:=5.0;
          end_if;
          if (pi_i_k<-5.0) then
                    pi_i_k:=-5.0;
          end_if;
}
```

Figure 3.24 - PI controller code sample.

The neural network used for simulation and for the Predictive PI Controller was implemented through the acquisition of a large group of data, representing most of the possible behaviors associated to the conceptualized use of the tank. That data is used to train the network via Levenberg-Marquardt back-propagation algorithm available in Matlab's neural network toolbox.

After the testing of several different trained neural networks, as seen in the Figure 3.25 and equation (3.5), the following structure was chosen. This network, on the entrance layer, holds information about three samples of the past regarding the tank level values ($y(k-1)$, $y(k-2)$, $y(k-3)$) and two regarding the actuation applied ($u(k-1)$, $u(k-2)$), one internal layer holding three neurons that use the hyperbolic tangent as an activation function and one neuron output layer that delivers the current predicted sensor value $y_{nn}(k)$. The W matrixes represent the weights that result from the network training and are the key factor to represent the model behavior. The B matrixes hold the bias values that serve purpose to calibrate the activation functions within the neural layers.

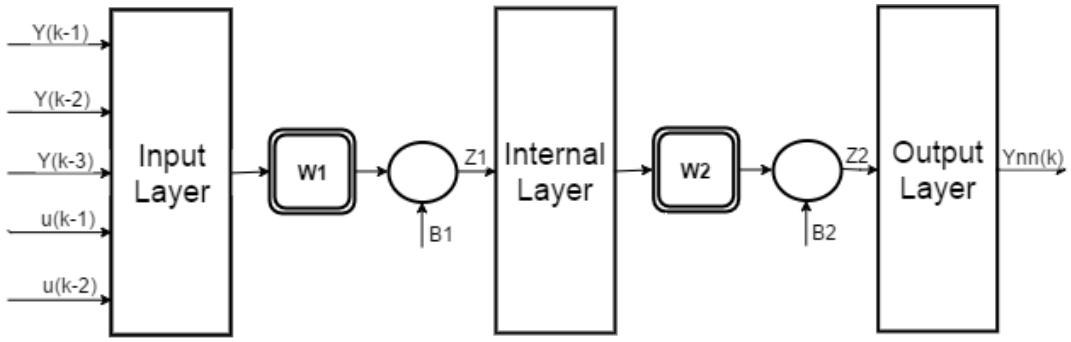


Figure 3.25 - Chosen neural network structure.

$$Z_1 = W_1 * [y(k-1)\,..\,y(k-3)\;u(k-1)\;u(k-2)]' + B_1, \tag{3.4}$$

$$Z_2 = W_2 * tanh(Z_1) + B_2 \tag{3.5}$$

$$y_{nn}(k) = \tanh(Z_2) \tag{3.6}$$

Where $W_1 = \begin{bmatrix} -35.44 & 21.33 & 12.23 & -1.71 & 2.1 \\ 0.15 & 0.08 & 0.08 & -0.002 & 0.0081 \\ -7.66 & 29.21 & -27.93 & 3.49 & 2.78 \end{bmatrix}$, $B_1 = \begin{bmatrix} 2.23 \\ -0.17 \\ -0.76 \end{bmatrix}$, $W_2 = \begin{bmatrix} -0.04 \\ 3.25 \\ 0.0027 \end{bmatrix}$ and $B_2 = 0.58$.

Due to the inability of the M340 PLC to support calculations involving matrixes or more advanced mathematical functions, such as hyperbolic functions, the neural network was implemented as observed in the Figure 3.26.

```
neural{

x1_11:=w1_11*y_sensor_k_m1+w1_12*y_sensor_k_m2+w1_13*y_sensor_k_m3+w1_14*u_motor_k_m1+
w1_15*u_motor_k_m2;

x1_21:=w1_21*y_sensor_k_m1+w1_22*y_sensor_k_m2+w1_23*y_sensor_k_m3+w1_24*u_motor_k_m1+
w1_25*u_motor_k_m2;

x1_31:=w1_31*y_sensor_k_m1+w1_32*y_sensor_k_m2+w1_33*y_sensor_k_m3+w1_34*u_motor_k_m1+
w1_35*u_motor_k_m2;

 z1_11:=(exp(x1_11)-exp(-x1_11))/(exp(x1_11)+exp(-x1_11))+b1_11;
 z1_21:=(exp(x1_21)-exp(-x1_21))/(exp(x1_21)+exp(-x1_21))+b1_21;
 z1_31:=(exp(x1_31)-exp(-x1_31))/(exp(x1_31)+exp(-x1_31))+b1_31;

  x2:=w2_11*z1_11+w2_12*z1_21+w2_13*z1_31;
  ynn_k:=(exp(x2)-exp(-x2))/(exp(x2)+exp(-x2))+b2;
}
```

Figure 3.26 - Code regarding the model prediction.

Lastly, concerning the received values from the water sensor, it does hold a nonlinear scalability to its real physical value meaning, therefore, besides the normalization of the range [3.83 to 9.32] Volt, there was also the need of using an approximate polynomial regression, in order to have an accurate normalized value of the water level within the tank, as seen in Equation (3.7).

$$y(k)_{scaled} = \frac{0{,}0051 * y(k)^3 - 0{,}1281 * y(k)^2 + 1.1582 * y(k) - 2.8088}{1000} \quad (3.7)$$

3.4 Supervision and Monitoring

The supervision and monitoring heavily relies in the use of the group of a controller and a graphical HMI. The role of the supervisor is to receive all of the main information coming from the Human-machine interface, conveyor and tank controllers. With that information it sets alarms and imposes the general work state coming from the HMI that all the other components have to adapt to. It is also the only component that directly communicates with the remote clients.

The Human Machine Interface or HMI is used as a high level graphic interaction hardware, which allows the user to observe and manipulate the whole system within the features it provides.

3.4.1 Operational Description

Concerning the supervisor controller, it essentially divides itself in the following operations:

- Alarm setting: there are six physical lights and two buzzes associated to the controller. Each buzzer symbolizes if the conveyor or the tank have encountered a failure. The lights hold information about the supervisor, conveyor and tank current mode, communication accessibility and stop state.
- Information propagation: has a large reserved memory that holds information about the main aspects of each component, with the goal of establishing information access to the remote clients and HMI.
- HMI support: many graphical tasks implemented were not possible if not by using the supervisor; such tasks were mainly animations and trigger screens.

Regarding the HMI, its function is to give to the user the full capability of visualization and interaction with all of the system, without having the need to physically interact or monitor the processes.

The screen programmed structure majorly divides itself between the manual and automatic mode, which are individually accompanied by failure, block, option choosing and help glossary screens.

The manual mode holds a very simple interface, where it can be observed and manipulate in real time both processes with ease (Figure 3.27).

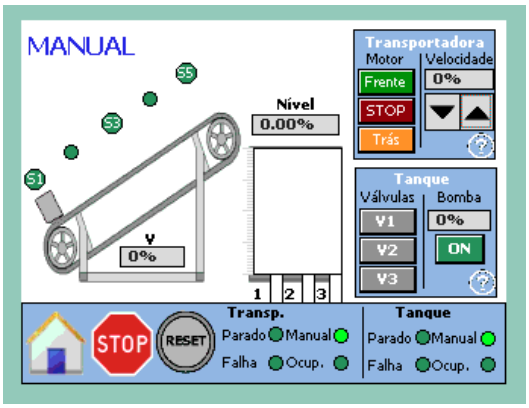


Figure 3.27 - Main manual mode Screen.

If in any case one of the processes is not communicating or is in stopped state, the interface will inform the user and will redirect into a manual mode where only the operational process can be viewed and actuated upon (Figure 3.28).

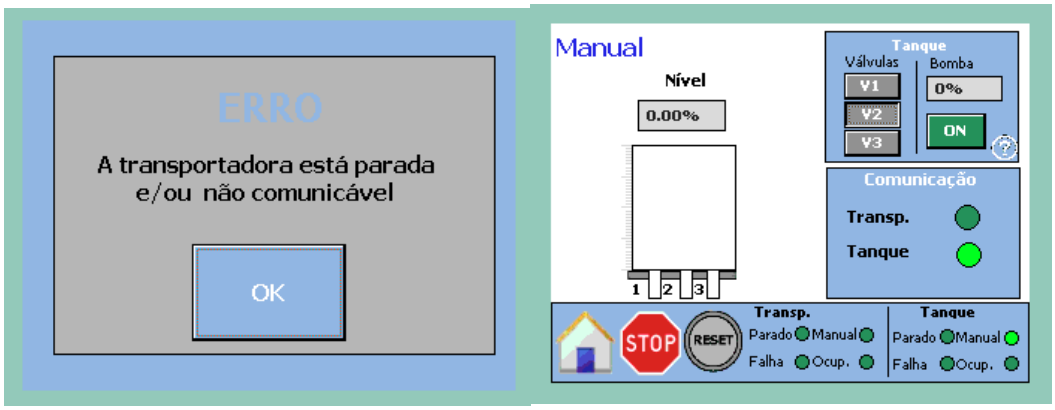


Figure 3.28 - Conveyor anomaly and consequent manual mode screen.

Concerning the automatic mode, it will only be able to be chosen if both processes are operational; if they are, the user will be forwarded into specification choosing.

As seen in Figure 3.29, there are several custom parameter. Within the HMI, it was defined as there being three kinds of parts:

- Part A: holds standard specifications regarding a smaller and more robust treated part, therefore, higher conveyor speeds, less treatment time, lower tank level and a sequential controller.

- Part B: holds standard specifications regarding a larger and more fragile treated part, consequently, lower conveyor speeds, more treatment time, higher tank level and a continuous controller.

- Part C: this part allows the user to implement its own desired treatment, allowing the customization of all the specifications offered by the tank and conveyor control systems.

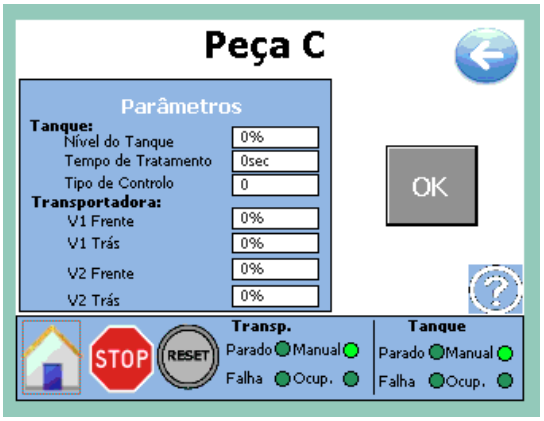


Figure 3.29 - Custom part specification selection screen.

Regarding the automatic's mode main window, as viewed in Figure 3.30, besides the observation of how the processes are behaving, it is also possible to:

let the system know when there is a part ready to be transported, jump the current treatment so a new control cycle can be initiated, application of simulated failures and change the part control parameters for the upcoming part. On a last note, it can also be viewed the current specifications being applied onto the part, and the future ones selected for the next.

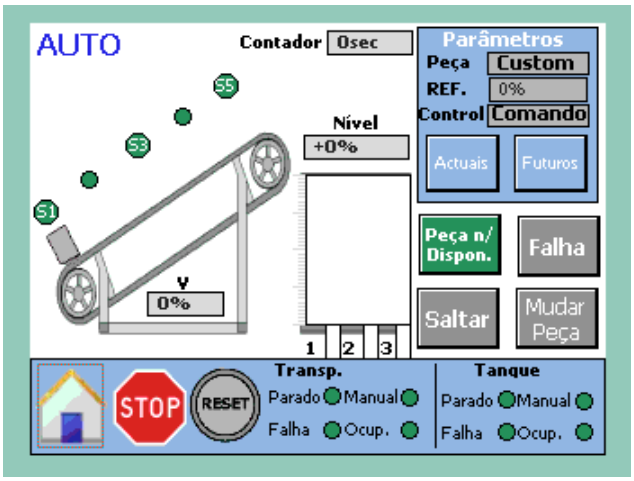


Figure 3.30 - Automatic mode screen.

Regarding the failure feature, it possesses the same graphical execution of the logic implemented in the process controllers: if a failure is detected a window will show up informing the user. It will wait a set period of time and, if there is no action taken, the system will globally block itself. If the user is present, he will be able to block the process that holds the failure, block the whole system or choose to go into a manual monitoring mode where he will be able to access the process behavior (Figure 3.31 and Figure 3.32).

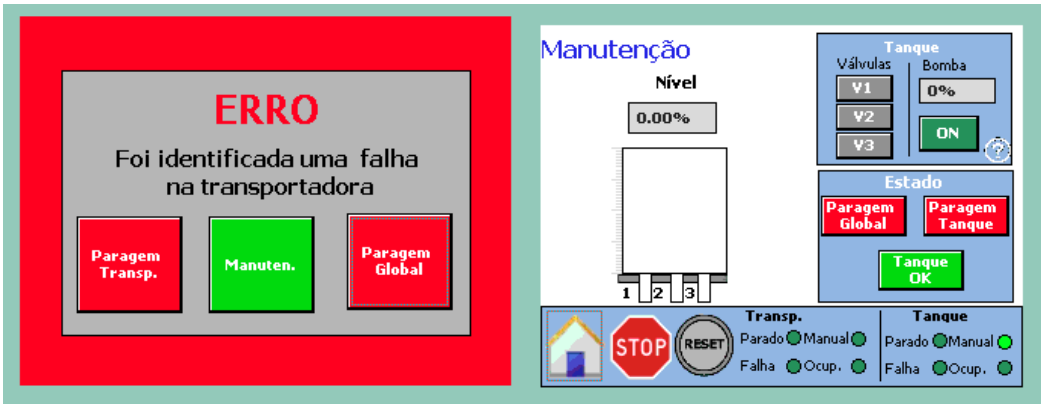


Figure 3.31 - Failure, monitoring mode screens.

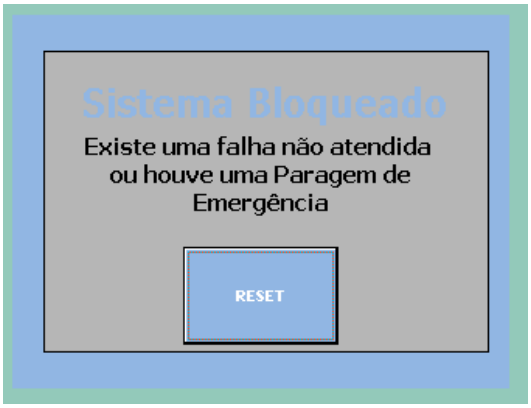


Figure 3.32 - Emergency block screen.

On a side note, there is also a simple help glossary accompanying every step of the graphical interface, as also a main access index (Figure 3.33).

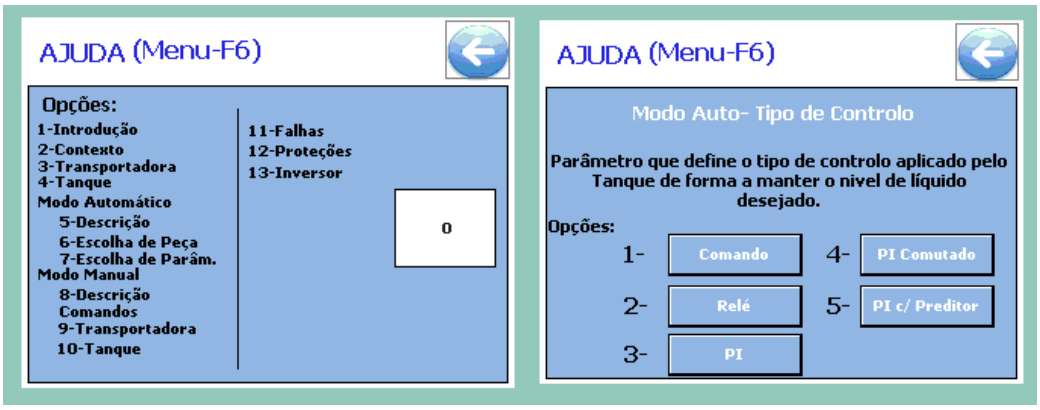


Figure 3.33 - Help screens.

3.4.2 Hardware Composition and Implementation

As seen in the Figure 3.9, the hardware used regarding this level were a couple buzzers, six physical lights, three buttons a controller and a human-machine interface

The controller used here was a Siemens s7-1200 and the HMI was a KTP600 Basic. The HMI allows for a very limited colored interface for industrial safe interaction because it can only support directly one controller. Therefore, there is the need of directing the information to the supervisor controller before being accessible in the HMI.

3.4.3 Software Methodologies

For the programing of the controller, it was used the proprietor software Step 7® and for the HMI the WinCC®, both being offered within a global tool software named Siemens TIA portal.

As it can be seen in the Figure 3.34, the software is composed by a simple graphical edition sector (a toolbox for the application of several visual objects) and an object properties sector where you can edit each graphical object used.

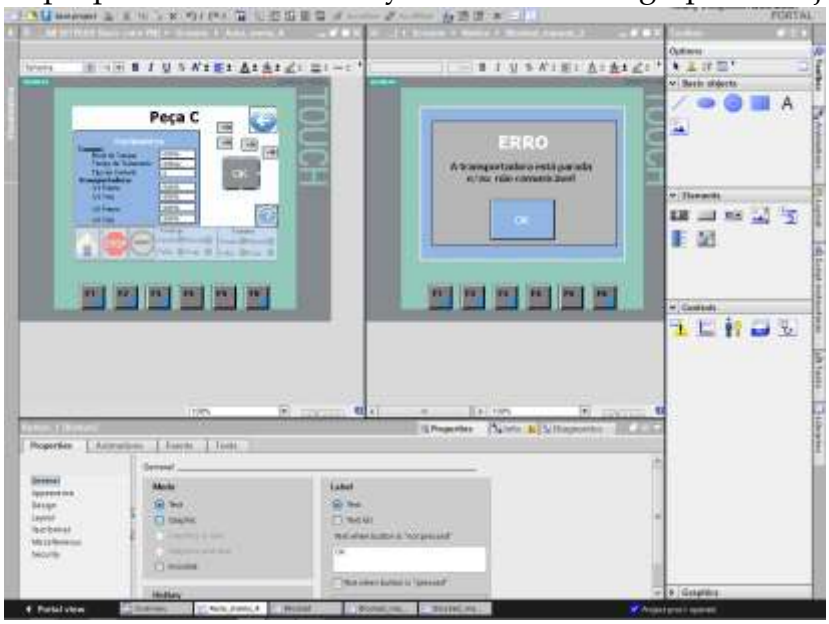


Figure 3.34 - Simatic WinCC/TIA Portal Software environment.

Both the controller and HMI do not own large or complex programming codes. The HMI programming is essentially a drag and edit kind of programming where the major concern was the esthetic and ease of monitoring. Concerning the controller, it holds very small tasks, tackling the different operational tasks mentioned in the sub-chapter 3.4.1.

One of the major details that must be mentioned was the well-functioning of the event triggered screens. In this project, they represent informational notice screens that pop-up when there is some kind of anomaly, particularly failure or an emergency stop performed within the system.

The way it works within the HMI is through associating a memory variable with an event on value change, so each time that variable changes value it forces the visualization of the alert window. Due this kind of event, there was the need to implement on the supervisor controller, a task that would indirectly associate the state variables with the event triggered variable, in order to avoid incorrect pop ups. The code exert within the Figure 3.35 below illustrates a simple commutation logic that supports the HMI failure triggered screens.

```
IF "failure_proc2"=1 THEN
  IF "fail_flag_proc2" = 0 THEN
    "fail_flag_proc2" := 1;
    IF "fail_screen_proc2" = 0 THEN
      "fail_screen_proc2" := 1;
    ELSE
```

```
            "fail_screen_proc2" := 0;
          END_IF;
        END_IF;

      ELSE
        "fail_flag_proc2" := 0;
        IF "failure_proc1" = 1 THEN
          IF "fail_flag_proc1" = 0 THEN
            "fail_flag_proc1" := 1;
            IF "fail_screen_proc1" = 0 THEN
              "fail_screen_proc1" := 1;
            ELSE
              "fail_screen_proc1" := 0;
            END_IF;
          END_IF;

        ELSE
          "monitor_ok_supervisor2" := 0;
          "fail_flag_proc1" := 0;
        END_IF;
      END_IF;
```

Figure 3.35 - Failure Trigger screen code.

Another important aspect of the controller support programming for the HMI was the animation of the interface conveyor. In order for the movement of the carrier to be visualized correctly, there was the need of creating a task that would manipulate the HMI translocation animation in conformity with the real information of the conveyor process.

The way the basic animation within the HMI works, is through specifying the beginning and end point of the translocation, and associating to it a scale of position values. Since there was the need of implementing a sensation of continuous movement within the image, the scale implemented had to be large enough so the increment of the position value would be slow.

Within the controller, it was implemented the following code exert, within Figure 3.36, that would take notice of the direction and sensor values coming from the conveyor process and either increment or decrement the animation position value depending on the conveyors direction, as well as forcing specific positions, depending on the conveyor sensors.

```
IF "conveyor_direction"=2  THEN
  "conv_move" := "conv_move" + 1;
END_IF;
IF "conveyor_direction"=1 THEN
  "conv_move" := "conv_move" - 1;
END_IF;
IF "l_switch_1_supervisor" = 1 THEN
  "conv_move" := 0;
END_IF;

IF "o_sensor_1_supervisor" = 1 THEN
```

```
    "conv_move" := 7500;
END_IF;
IF "o_sensor_2_supervisor" = 1 THEN
   "conv_move" := 15000;
END_IF;
IF "o_sensor_3_supervisor" = 1 THEN
   "conv_move" := 22500;
END_IF;

IF "l_switch_2_supervisor" = 1 THEN
   "conv_move" := 29000;
END_IF;

IF "conv_move" > 29000 THEN
   "conv_move" := 29000;
END_IF;
IF "conv_move" < 0 THEN
   "conv_move" := 0;
END_IF;

IF "conv_move" <> 0 THEN
   IF "conv_blink" = 1 THEN
      "conv_blink" := 0;
   ELSE
      "conv_blink" := 1;
   END_IF;
END_IF;
```

Figure 3.36 - Code exert for the Conveyor animation task.

3.5 Open Protocol Server and Communication

The Open Protocol Communication Server, or OPC server, is the main foundation for the transmission of information within this system. It works as a translator between devices that use different communication protocols, allowing a higher degree of interoperability between components.

The main reason for the need of an OPC server in this project, was due to the fact of there being different manufactured controllers with their own proprietary communication protocols which would make it impossible to establish data trade without the OPC server. A second reason was so it would be possible to transmit low level process information coming from the PLC controllers to higher level remote applications.

3.5.1 Operational Description

As mentioned in sub-chapter 3.1, it was opted to implement a centralized communication structure around the OPC Server where most of the components,

apart from the remote clients (that only read information from the server), work both as masters and slaves, depending on the information that is traded.

As seen in the Figure 3.37, the OPC server has to work with multiple communication protocols chosen depending on each controller's communication features and the OPC software protocol drivers available.

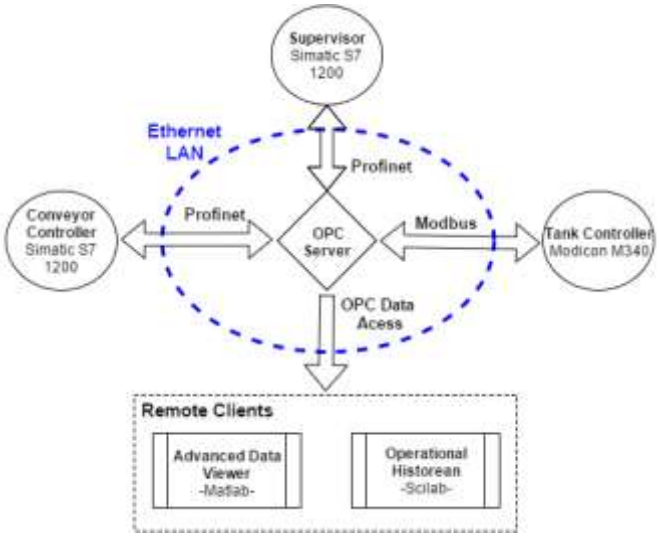


Figure 3.37 - Communication structure scheme.

The way the general OPC communication system works is based on device configuration, memory linking between devices and consequent OPC protocol driver application.

Within the OPC server, it is simply appointed the type of device, its internet protocol address and selection from the available protocol communication drivers. After that, the declaration of the data to be transmitted is issued and there is the need to define the memory address used within the controller, the type of data, and the desired scan rate.

When all the controllers and their associated variables are configured, it is needed to apply the linking function to correlated variables between devices (Figure 3.38), so the trade of information is possible. In other words, connect the memories of the data from the sender to the receiver.

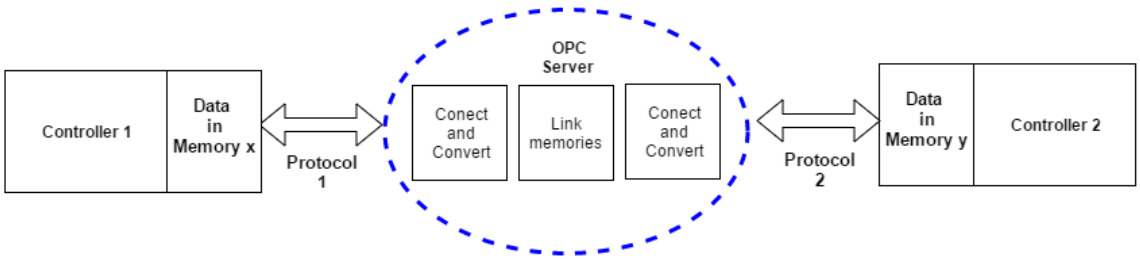


Figure 3.38 - Memory linking between Controllers.

The case of the remote clients is a bit more complex (Figure 3.39). The information does not reach the application software directly; instead, the data is sent from an OPC server to an OPC client; from that, the application proceeds to request the information. Regarding the information that is traded between the system components, it is detailed illustratively in the section of the Appendix D.

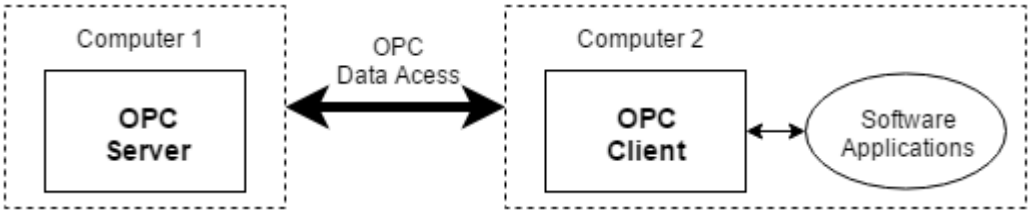


Figure 3.39 - Illustration of remote information access.

3.5.2 Software Methodologies

For the Open Protocol Communications it was used the KEP Server EX. It offers a vast array of protocol drivers for numerous PLC controllers and software, its functionalities rely heavily on device configuration and tag creation.

The way the different variable memories are linked is through the use of the advanced tag feature. This concept portrays the application of adjacent functions to a tag, such as complex, average, minimum calculations.

Regarding information acquisition, on what concerns the PLCs, there is no need to apply any kind of instructions to read or write from the server, since the OPC Server will update the data within the memories directly. In the case of the remote clients, however, depending on the application software used, there is the need to apply function libraries that are responsible for issuing a request, receiving and processing the information.

Concerning update rates within the server, controller related data was attributed a scan rate of 100 milliseconds; and less important visual information for high level monitoring holds 200 milliseconds rate. The software allows for lower scan rate specifications up to 50 milliseconds; however, it was opted not to choose lower values due to the fact that most of the commercial Ethernet networks have minimal scan rates of 100 milliseconds.

About the specification of the correct address of the controllers variables within the server, we must note that they are dependent of what type of protocol is being used. In the case of the Profinet TCP/IP protocol, the address is the one declared within the controller; in the case of Modbus TCP, the address depends on the type of variable declared on the controller. For simplicity and compatibility purposes, it was made sure that the variables being transmitted or

received in the Tank controller were all of the Word type, therefore, having a correspondent server address of 400000+controller memory location.

3.6 Remote Clients

With the effort of allowing this system to have a higher degree of accessibility and more complex data processing, transmission was established with remote computers that, with a varied array of software, could take the control system even further.

In this project, two clients were implemented: an operational historian and an advanced data viewer. This applications constitute read only events, i.e.: do not influence the main system in anyway; although, the structure and software tools could allow remote control applications.

3.6.1 Operational Historian

The operational historian is used essentially as an information back up and log for all of the transmitted data relating to the supervisor, tank and conveyor. The need for the implementation of this client was to tackle the lack information registry for long periods of time, the controllers within the system have their memories constantly updated, therefore offering no data record features.

3.6.1.1 Operational Description

As seen in the Figure 3.40, this client performs a cycle where it logs into the OPC client, retrieves the desired group of data and writes into the console a log of the main state information about each component. Afterwards, it saves the data separately into individual files regarding the conveyor, tank and supervisor, ending on a memory check.

Due to performance and memory resource constraints, it was established a limited array size to each variable. When that array is full, the historian will perform a global save, reset the arrays and begin the data saving onto new files. The higher the size of each array, the longer the period of recording between memory resets and higher the client initialization setup time. In this project, it was opted for an array size of 7200 and a sample time of 1 second, which amounts to 2 hours of recording between resets.

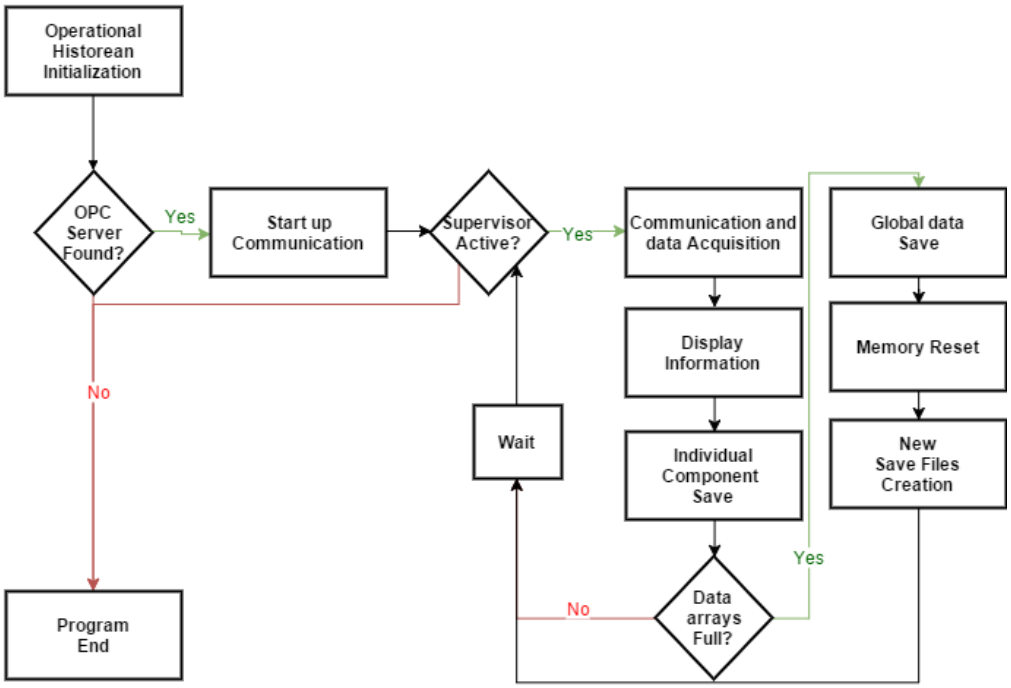


Figure 3.40 - Instruction logic of the historian client.

3.6.1.2 Software Methodologies

For this client, the software chosen was Scilab®. It is an open source numerical oriented programming environment. The reason for its use comes mainly from the fact of possessing easy to use function libraries for OPC communication.

One of the most important aspects of the software implementation is how the client acquires the information from the OPC Server. For that task to be possible it was necessary the use of an OPC toolbox add-on for the Scilab® software made by Zhe Peng. The application is as follows: (i)start the OPC server browsing, (ii)connect to it, (iii)create an OPC group that holds an array with the structure paths of the items you desire to read from and (iv)perform an OPC item read every time an information update is needed. The code relating to this routine can be viewed in Figure 3.41.

```
opc_server_name = 'Kepware.KEPServerEX.V5';

opc_group_name = 'group1';
item_read=zeros(2,1);

item(1) = 'Client_Read.Tank.proc2_active';//1=is comunicating
item(2) = 'Client_Read.Tank.failure_proc2';//1=fail
counter1=0;


found_server = opc_server_browse();
```

```
if size(found_server,'*') <> 0 then
    opc_connect(opc_server_name);
    opc_item_browse();
    opc_add_group(opc_group_name);
    for counter1 = 1 : 2
        opc_add_item(item,counter1);
    end;
    while (program_running)
        item_read=opc_item_read(2,'i');
        Tank_active= item_read(1,1);
        Failure= item_read(2,1);
    end;
    opc_disconnect();
end;
```

Figure 3.41 - OPC communication code example.

The file composition is a Scilab® data file that holds an array of samples. Each sample contains a time stamp and data structures with the information obtained at that instant.

On what concerns the log presentation, as seen in the figure within Appendix section C, it was conceptualized to be simple and it was divided into three groups: display of system states, I/O process values and work mode related information.

3.6.2 Advanced Data Viewer

The advanced data viewer purpose is to allow a more detailed and remote observation of the tank controller system, by applying different graphical methodologies concerning failure monitoring. Its necessity came from the high complexity and low performance associated to programing a more elaborate monitoring system at the controllers/HMI level.

3.6.2.1 Operational Description

This remote client works similarly to the operational historian (Figure 3.42), by connecting to the OPC server first, and retrieving data (system state information and the complete tank data). Afterwards, if the supervisor is active, the advanced data viewer will check the tank's mode. In the case of being in manual mode, it will simply present graphical information about I/O values of the tank, along general system state information being presented in the form of a log.

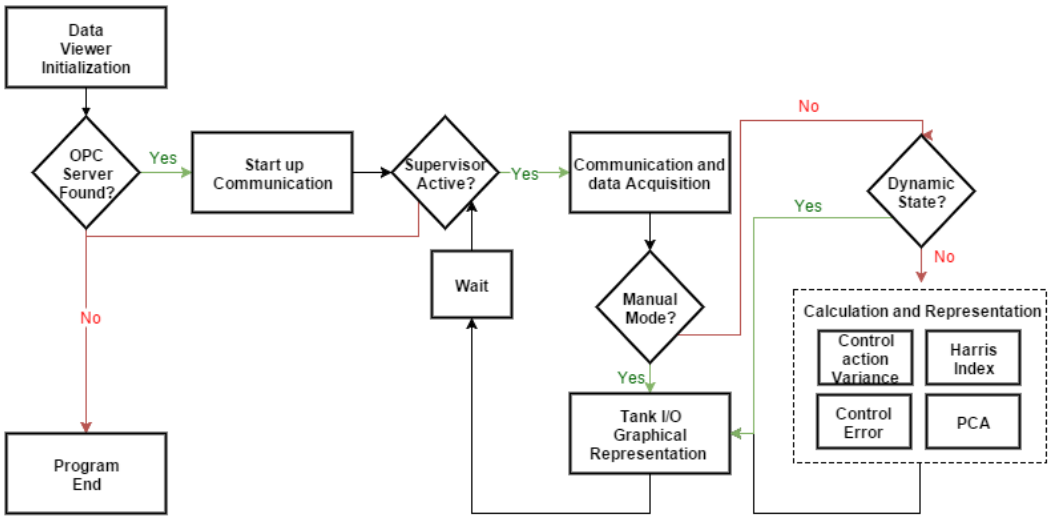


Figure 3.42 - Operational logic scheme of the data viewer client.

When in automatic mode, it will add the graphical representation of the following features:

- **Control error** :
  Difference between the specified tank level and the current value being read through the sensor.

$$e(k) = |r(k) - y(k)| \tag{3.7}$$

- **Control action variance**:
  Calculates the average variation for the control signal applied onto the water pump, allowing to deduce its stability.
  It is based on a sample sliding window of size N and returns the average value for the squared subtraction between each control action sample and the sliding window's average value.

$$\sigma^2(k) = \frac{1}{N} * \sum_{i=k-N}^{k} \left(u(i) - \mu\big(u(k-N:k)\big)\right)^2 \tag{3.8}$$

- **Normalized Harris index:**
  Indicator proposed by Desborough and T.J.Harris [37], that evaluates the controller performance with minimal knowledge about the process behavior. Its value ranges between 0 and 1, where 0 indicates a good performance and 1 a bad performance.

$$I_{Harris}(k) = 1 - \frac{\sigma_{r_e}^2(k)}{x_e(k)} \tag{3.9}$$

  In which $x_e(k)$ represents the mean squared error of the control error and $\sigma_{r_e}^2(k)$ represents the variance concerning the residue between the control error and a predicted error.

$$r_e(k) = e(k) - \hat{e}(k) \tag{3.10}$$

The predicted error was based on a simple auto-regression model AR(n,b) which is represented by the following equation:

$$\hat{e}(k) = a_0 - a_b * e\big(k - (b+1)\big) - \cdots - a_{b+n} * e(k - (b+n)) \tag{3.11}$$

- **Principal component analysis**:
  It is a statistical method that orthogonally transforms a set of initial correlated variables into a new uncorrelated linear combination of variables (Figure 3.43). This new variable set is defined by principal components that allow further simplicity by variable set reduction and portrayal of the main variation characteristics.

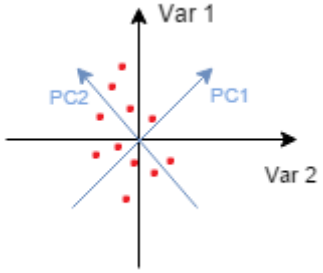


Figure 3.43 - PCA graphical representation [38].

The deduction of the principal components was done through the Singular Value Decomposition method (SVD) [39]. This method considers a matrix X_p of order N×M. N is the number of past samples and M is the number of initial variables. Knowing that the covariance of the matrix X_p is defined by:

$$S = \frac{X_p^T * X_p}{N - 1} \tag{3.12}$$

The PCA decomposes the matrix X_p in the sum of the product between the principal component values or score vectors and the process's characteristic matrix or loading matrix, t_i and P_i respectively. It represents the number of principal components and E represents a residual matrix.

$$X_p = \sum_{i=1}^{I} t_i * P_i^T + E \tag{3.13}$$

Meanwhile, P_i also represents the Eigen vectors of the matrix of covariance, hence the following equation:

$$S * P_i = \lambda_i * P_i \tag{3.14}$$

Where λ_i are the Eigen values associated with the loading matrix. Based on equations (3.12) and (3.13), the principal components are calculated with the following equation:

$$T = X_k * P \tag{3.15}$$

The vector T is the score vector, and it holds the new transformed values and X is the matrix that represents current values of the initial variable set and P is the loading matrix.

In this project it was used as a failure detection observation means, where to each work-state, the principal components would position themselves in a specific localization; in case an unexpected behavior happened, there would be a deviation from that localization thus pointing to a fault. Illustration in Figure 3.44.

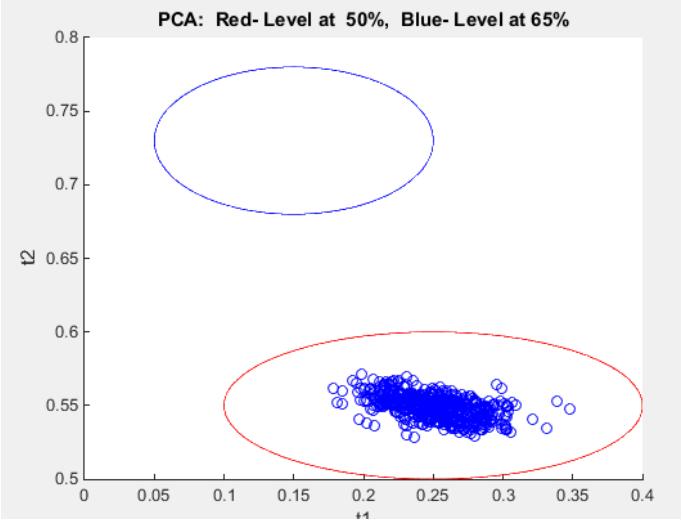


Figure 3.44 - PCA results for controller at nominal work state.

3.6.2.2 Software Methodologies

For this client, the software used was Matlab®, which is a multi-paradigm numerical computing environment. The reason it was chosen in this project was due to the ease of implementation for the features initially conceptualized.

Regarding the communication with the OPC Server, it is very similar to the operational historian logic, through the use of the Matlab® OPC toolbox code functions, thereby being started by connecting to the server and then, creating a group of items that hold the path to the item within the server and then read periodically.

One important aspect of the software implementation is the error prediction model for the performance index calculation. For the error to be predicted

correctly, the model parameters must be calculated. This was done through formulating the equation (3.11) in terms of a vector regression, as observed on equation (3.16), whom model parameters (3.17) are calculated during the control period.

$$\hat{e}(k) = \varphi^T(k) * \theta(k) \tag{3.16}$$

$$\theta(k) = [a_1 \dots a_{b+n}\ ]^T \tag{3.17}$$

$$\varphi(k) = [1 - e(k-b) - \cdots - e(k-(b+n-1)]^T \tag{3.18}$$

For the calculation of the model parameters vector $\theta(k)$, the mean squared algorithm is used and it is the cyclical execution of the following equation:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k) * \varphi(k) * \varepsilon(k) \tag{3.19}$$

Where P is the covariance matrix and ε is the model error, which are represented by:

$$P(k) = \frac{P(k-1)}{\lambda}\left[M_{identity} - \frac{\varphi(k) * \varphi^T(k) * P(k-1)}{\lambda + \varphi^T(k) * P(k-1) * \varphi(k)}\right] \tag{3.20}$$

$$\varepsilon(k) = y(k) - \varphi^T(k) * \hat{\theta}(k-1) \tag{3.21}$$

Concerning sample times, it was chosen 0.2 milliseconds wait period for each cycle. Also, the automatic monitoring methodologies were given a 30 seconds delay after the start of each controller, so that the dynamic transition would be ignored.

As seen in Figure 3.45, these are the interfaces that appear when in manual or automatic mode. The reason for the existence of a manual and automatic mode I/O plot is so the user can directly see at which instances each mode was activated and its respective I/O values since the beginning of the client operation.

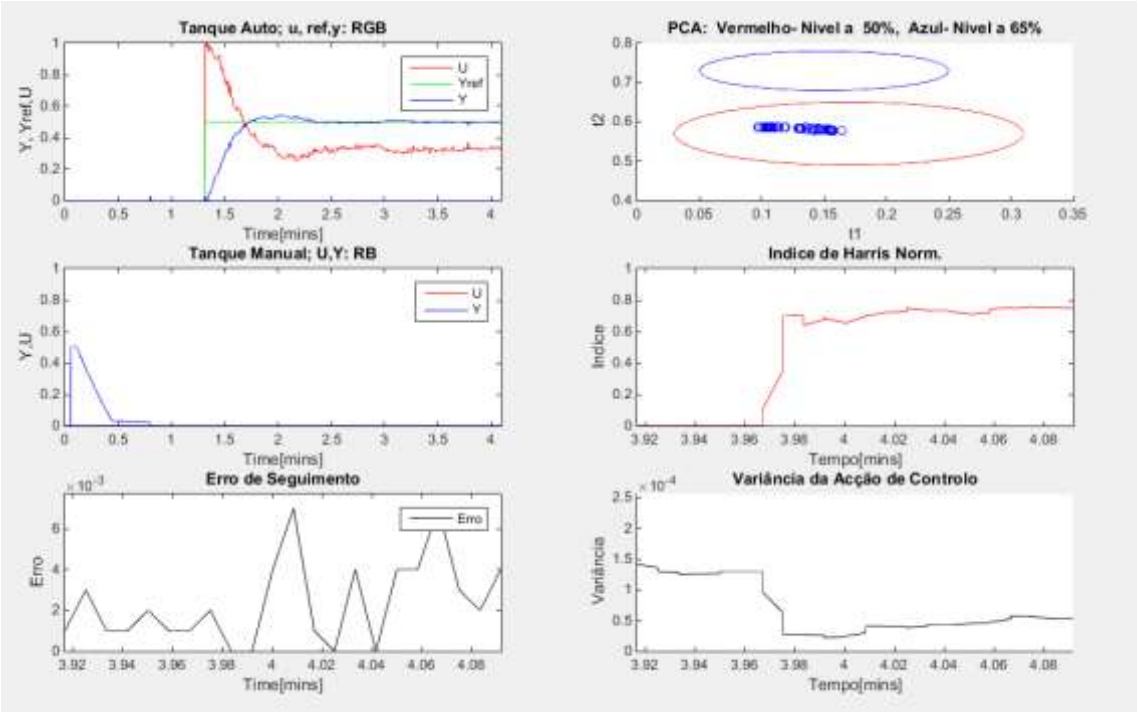


Figure 3.45 - Viewer monitor interface.

4. Experimental Results

This sub chapter will cover the main aspects tested that were important enough and might present a significant risk factor to the overall well-functioning of the system.

In sub-chapter 4.1, the topic of the conveyor process will be approached, focusing on the sensor limitations and max travelling speeds.

Within sub-chapter 4.2, it will be shown how the tank's water level sensor behaves when the water is being inserted into the tank and how vibrations influence the sensor.

On sub-chapter 4.3, it is illustrated the general testing of the closed loop controllers implemented within the tank system, comparing the different test indicators to compare the controllers between each other.

Finally, the sub-chapter 4.4 will go over the communication and point out the main risks and potential consequences of there being malfunction in data transmission.

4.1 Conveyor Testing

Having the conveyor speed limited to a maximum of 19.8 rpm, and considering that it remains constant for the transportation task, from one side to the other, it takes a minimum of 7 seconds to travel the entirety of the conveyor belt. Between direction changes, there is a wait time of 10 seconds to assure the repositioning of the transported/to be transported object. On what concerns the motor stopping its movement when the carrier reaches the limit sensors, trough testing, it was deduced that if the defined speed is above 50% of the before mentioned max allowable speed of 19.8 rpm, the conveyor will stop its movement after the carrier passes the sensor and ends its range of activation, which consequently will force a carrier calibration task in the following transportation.

About the optical sensors, since they both work as transmitters and receivers, there was the need of the conceptualization of a reflector to attach on to the carrier. After many experiments, it was chosen the combination of a part of foil placed within a red light reflector.

Figure 4.1 illustrates how the main variables react while the controller processes the automatic task of carrying a part to the next station and returning to the initial position, as explained in the sub-chapter 3.2.2.

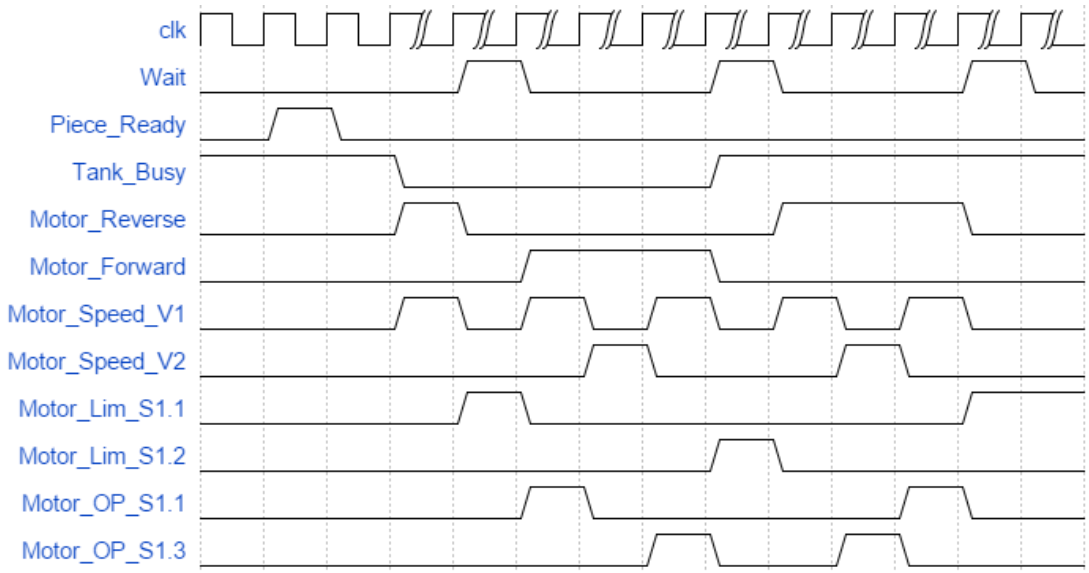


Figure 4.1 - Conveyor timing diagram.

4.2 Tank's Sensor Testing

This section is dedicated to illustrating the behavior of the sensor while on steady water and as the water pump is on. The need for this test comes from the existent noise within the tank infra-structure whenever the water pump is activated. This noise consequently influences the controller's performance, especially on its output variance.

The first test illustrates the values originated by the sensor when there is no action applied on to the tank. The second test represents the values read from the sensor when the inflow and outflow of water are identical while the water pump is on. The duration of both tests was 40 seconds and the water level used for reference was approximately 50% (in the scale of 0 to 1).

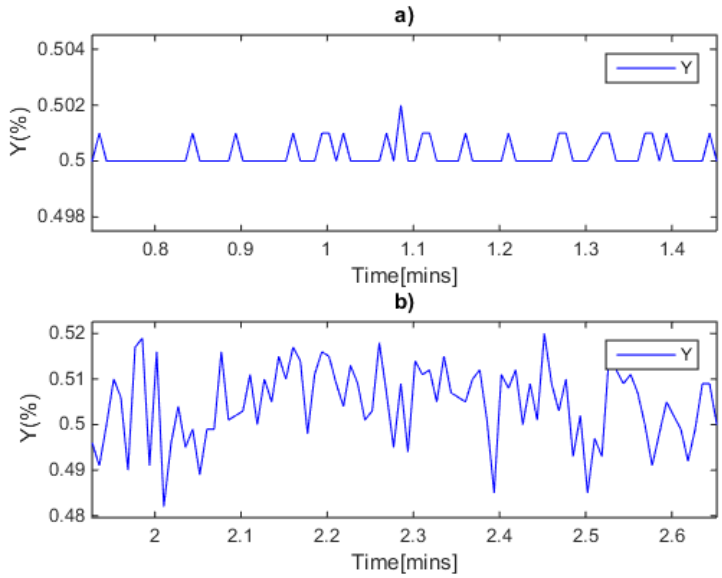


Figure 4.2 - Sensor Testing.

In the Figure 4.2, it can be observed a very significant difference between the values read when the water is steady (test a) and the values read when the water pump is on (test b). The maximum deviation within test a) is of 0.2% and the variance associated to it is of 0.0022; on test b) worse values were obtained, with a maximum deviation of 2% and a variance associated of 0.77. This obvious difference caused by the vibration of that water pump, makes the controllers performance regarding their output actuation significantly less stable.

4.3 Tank's Controller Testing

In this section, it shall be presented the experimental testing of the closed loop control methodologies implemented within the tank system (sub-chapter 3.3.2) with the objective of showing their general behavior and performance.

The methodologies presented are the PI controller, Switched PI and PI with model rectification. Relay and command controllers are not presented due to the fact that one is simply a static controller that only places the water at the specified level and closes the valves, and the other is for less demanding systems with robust actuator, where computational limitations are prioritized against controller precision and efficiency.

The test scenario for all the methodologies considered will be the control of the tank level at 50% in nominal work-state (with the scale of 0 to 1). The data illustratively presented for analysis will consist of sensor and controller outputs, set-point errors, mean-squared errors and control output variance. Furthermore, we will study the overshoot, rising and settling time. The duration of the test will

be 200 seconds with a considered dynamic state interval of 72 seconds (which some of the variables mentioned above will ignore).

No failure testing is performed because the system enters in failure mode immediately in case the controller is unable to keep the water level at the specified value for a long enough interval of time.

4.3.1 PI Controller

As it can be observed from Figure 4.3, the application of a tuned PI controller to a global set point interval, holds a considered good performance for a conceptualized slow work state. It holds a rising time of 99 seconds and it stabilizes around a settling time of 140 seconds, with an over shoot of 7.1% from the 0.5 set point value. Moreover, an average set point error of 0.0047 or 0.94%, a mean square error of $3.3571 * 10^{-5}$ and an actuation variance of $1.2261 * 10^{-4}$. These values are small enough to be acceptable in the general function of the control task.

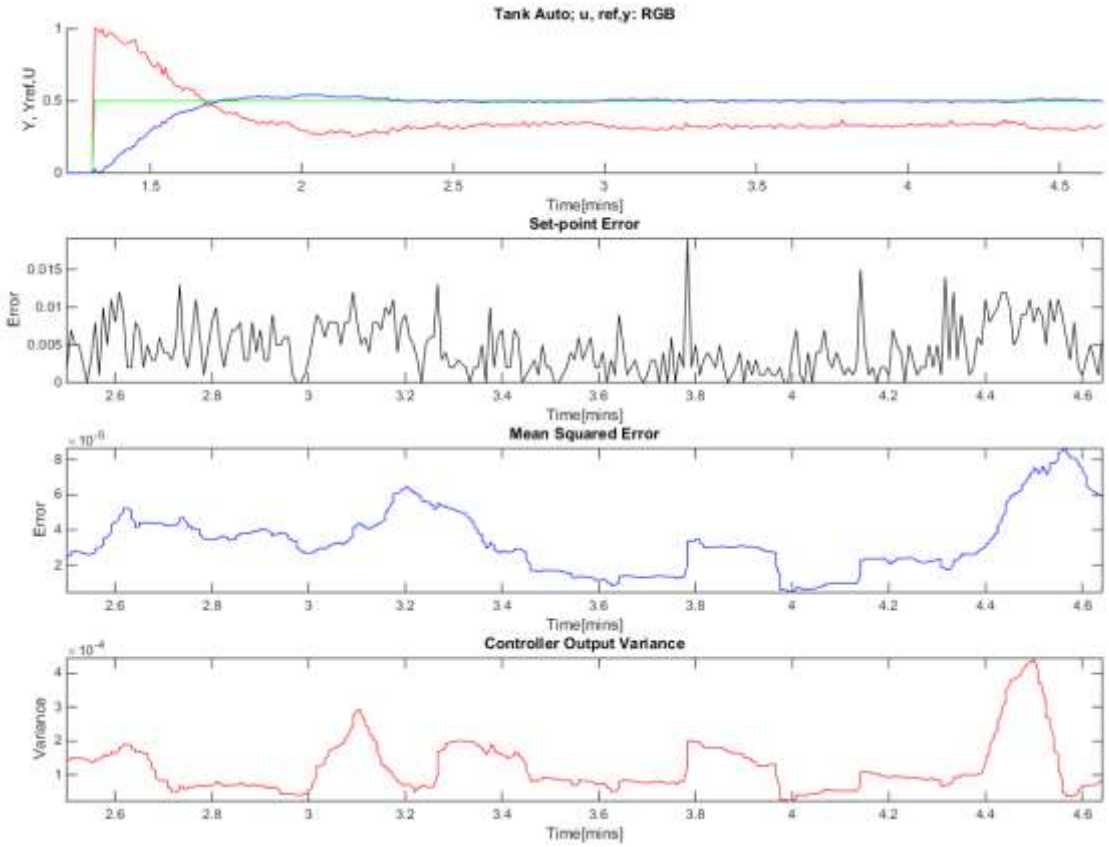


Figure 4.3 - PI Controller behavior.

The only observed problem is the lack of value convergence from the error and variance indicators along the time of control application. The main reason

for this to happen is related with the innate noise associated to installation vibrations and sensor quality.

4.3.2 Switched PI Controller

As it can be observed in Figure 4.4, the application of a Switched PI controller that holds different tuning parameters, depending on what the area of effect tank level is, shows slightly better results compared to the former controller. It holds a rising time of 78 seconds and it stabilizes around a settling time of 118.8 seconds, with an over shoot of 2.28% from the 0.5 set point value. In addition, it holds an average set point error of 0.0025 or 0.5%, a mean square error of $1.3271 * 10^{-5}$ and an actuation variance of $1.9865 * 10^{-4}$. These values are small enough to be acceptable in the general function of the control task.

Comparing to the former controller, the lack of value convergence at small decimals is still present but holds lower errors and a higher actuation variance. This is the product of the application of more fine-tuned parameters for smaller work state intervals. The higher actuation variance stems from the fact that the parameter dimensioned were picked in order to minimize time requirements and overshoots.

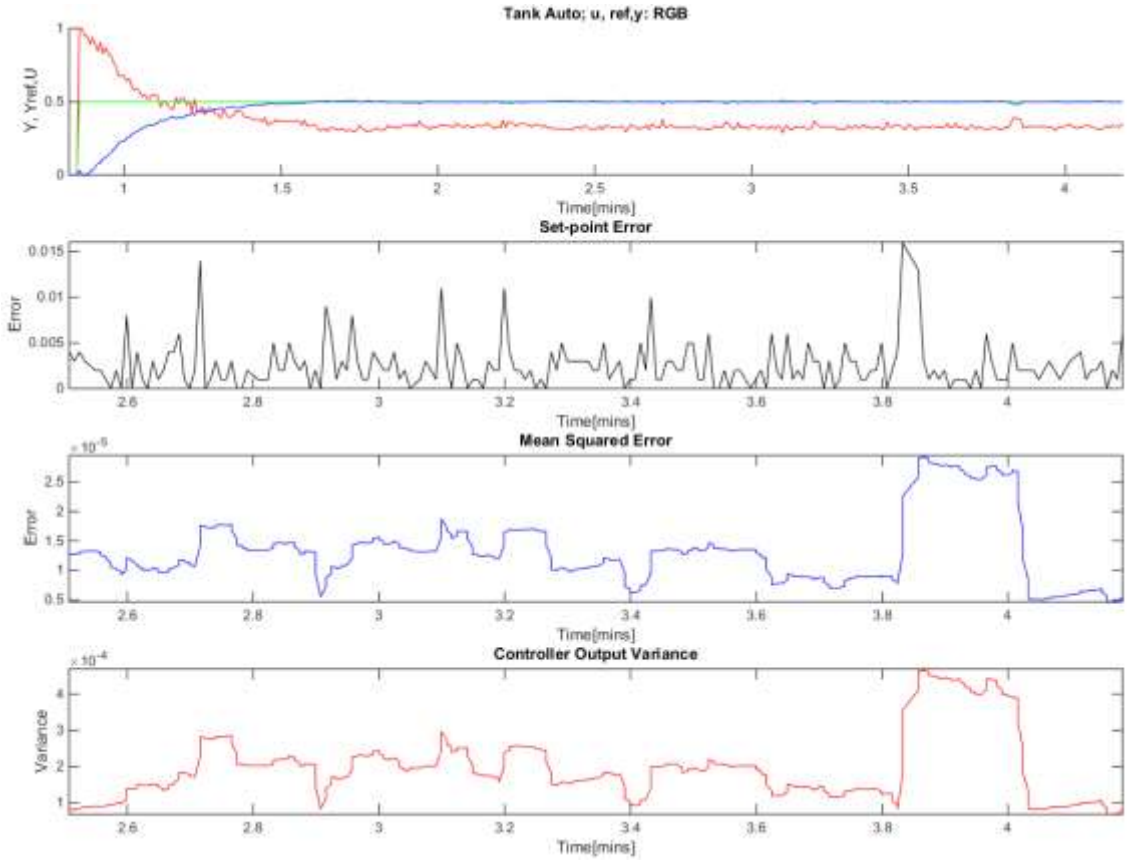


Figure 4.4 – Commuted PI Controller behavior.

4.3.3 PI Controller with Model Rectification

As it can be seen in Figure 4.5, the addition of a neural model to rectify the first PI controller presents some interesting results. It holds a rising time of 74.4 seconds and it stabilizes around a settling time of 117 seconds, with an over shoot of 5.05% from the 0.5 set point value. Also, it holds an average set point error of 0.0034 or 0.67%, a mean square error of $1.8523 * 10^{-5}$ and an actuation variance of $4.227 * 10^{-4}$. These values are small enough to be acceptable in the general function of the control task.

Comparing to the former controllers, it does hold lower time requirements but presents a much higher actuation variance. Besides, although it presents slightly lower error and overshoot values, compared to the regular PI, the Switched controller still performs noticeably better on those parameters.

The variance issue comes from the fact that the value predicted by the model is not precise enough to sustain rectification that avoids control oscillation.

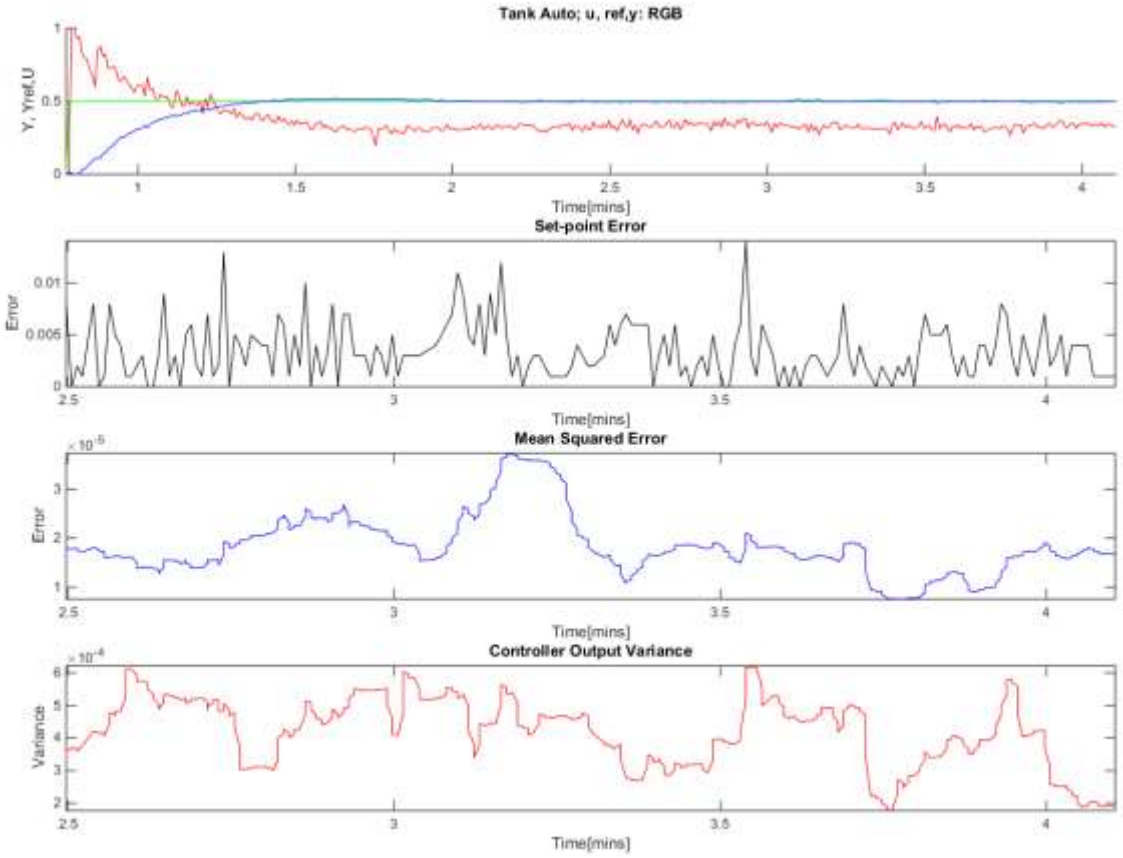


Figure 4.5 - PI Controller with model rectification behavior.

4.3.4 Comparison between Controllers

The five implemented controllers can be chosen depending on the goal that the user holds in mind. If there is no need to renew the liquid, the command controller will suffice. On the other hand, if the process is robust, precision is not a must, and the automation controllers have very limited processing capabilities, then the relay controller is the one to choose.

If the former two are not good enough then any choice of the different PI configurations is valid. The best overall performance is presented by the Switched PI. If time requirements are the number one priority, then, at the sacrifice of processing capability, the PI with parallel model is the one to go for. If there is only the need of an overall decent controller that is reliable in most of the areas to consider, then the normal PI configuration is the indicated.

4.3.5 Side notes relating to Remote Client Methodologies

As mentioned in sub-chapter 3.6.2, two additional methodologies, a PCA and a performance index. Relating to the PCA as long as the nominal work state regions are tested and calibrated, it will always allow for an exact and accurate visual means of deducing an anomaly. On the other side, the performance index implemented, due to it, on one side, being a very sensitive method (it is supposed to be) and, on the other side, the tank sensor being very noise influenced, leads onto very unstable results. In a perfect process, the index would start of being the worst value possible (being 1) and it would steadily converge to 0 (optimal performance value).

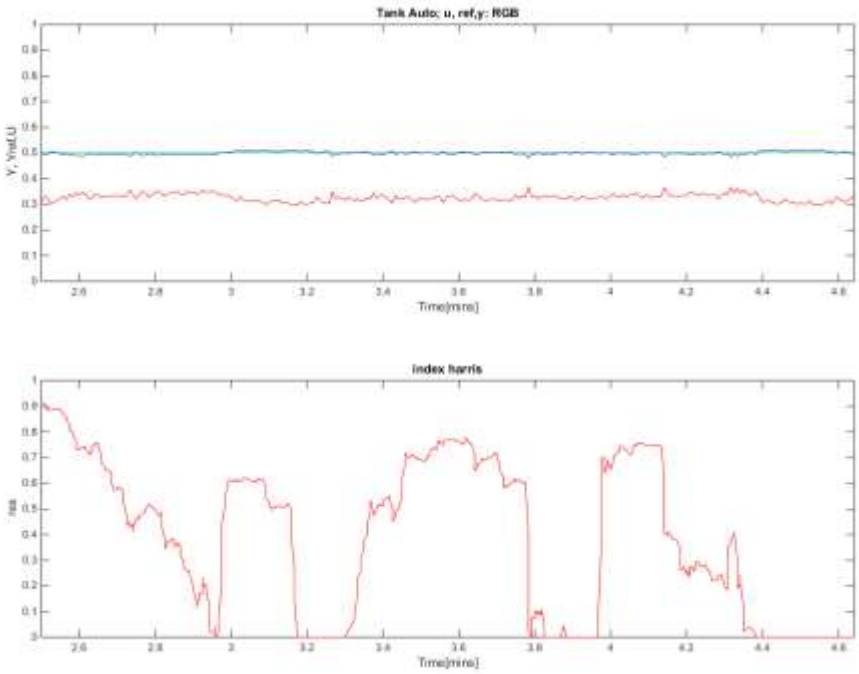


Figure 4.6 – Harris index results.

Due to the facts mentioned supra, it allows for slight control error oscillations which highly influence the method, making it vary often between 0 and 0.8. It can be observed in Figure 4.6 the behavior of the index for a PI controller at 50% tank level set-point after the dynamic state has ended.

4.4 Communications

Relating to communications, there is not much to say, the OPC server regulates all of the communications and in a general way, it asserts the 100 milliseconds scan rate for all the transmitted variables, which is the general minimal value permitted on regular Ethernet connections (50ms in some cases).

The most important risk factor is the Ethernet connection itself. If an anomaly happens, it puts to risk all the communication within the system. Nevertheless, since the system was built in a distributed way, the worst case scenario is the process controllers being in halt state waiting for a part to arrive or for the other process to be available. After 5 minutes of no connection, the server will de-activate the controllers and supervisor communication flags, which will lead both process to go into manual mode until the communications are reestablished.

5. Conclusions

In this chapter, the general conclusions of this dissertation are presented in sub-chapter 5.1. In sub-chapter 5.2, it is mentioned the honor associated to the implementation of this project .A proposition for future research in distributed systems in the control and automation areas is pointed in sub-chapter 5.3.

5.1 General Conclusion

The initial idea was to implement a personal interpretation of a distributed control system, and, taking everything into account, it can be viewed as a successful project.

A lot of the project was built based on unused material within the automation laboratory, therefore, shifting the focus of a more efficient and specific system conceptualization to the notion of making the best adaptation of the initial idea with the available resources at hand.

Regarding struggles of implementation, due to the fact of there being a lot of intersections between different natured equipment and software, the execution phase was highly susceptible to very unique glitches and bugs. This consequently led to less efficient and unconventional methods of implementing certain aspects, especially in the HMI device features.

Concerning the global system functioning, the main factor that must be pointed out is the time limitation in the scan rate of the information transmitted. In general, industrial automation systems have to be very strict with time requirements, making real time features a must on such systems.

In this dissertation a regular commercial Ethernet connection was used, thus making those time requirements very low. Realistically, they should present values around 0.01 milliseconds and instead, are limited by 50/100 milliseconds.

The trade off in time requirements allowed for a more versatile and open system platform, making the information horizontal to every future application that is able to retrieve information from an Ethernet based network.

In conclusion, this system represents a possible viable implementation of a distributed system that controls slow manufacturing processes and incorporates all of the main areas within modern automation systems, namely, in the pickling contextualization. It can also be viewed as a didactic means of illustration to new students not aware of the equipment and methodologies that can be involved in industrial manufacturing systems.

5.2 Awards and Honors

This project entered in the contest "Prémio Nova Geração 2015" organized by Siemens, and it was acknowledged as one of the finalists.

The contest had the main goal of challenging future engineers to develop automation projects that further improve the Portuguese Industrial sector. Its jury committee was composed by members of Siemens, CIP, Cotec, Cadflow and "Ordem dos Engenheiros".

5.3 Future works

According to the experience attained from the study and execution of this dissertation, a proposition for a distributed system with some of the following characteristics is suggested:

- Wireless technology focused either between the system components as also between sensors/actuators and process controllers;
- Use of smart product identification and manufacturing via RFID based technology;
- Execution of higher complexity remote clients that influence the global system;
- Application of communication methodologies that present higher real-time requirements and higher information accessibility;
- Implementation of significant information security standards.

Bibliography

[1] Tesla Assembly line, digital image. Available in https://www.teslamotors.com /blog/inside-tesla-060512, [Online]; Consult: 2015-July-1

[2] Adam Robinson, "Industrial Automation: A brief History of manufacturing", Available in http://cerasis.com/2014/10 /22/industrial-automation/, [Online]; Consult: 2015-July-1

[3] Peter Farkas, "Hard Vs. Soft Assembly Automation". Available in http://www.agi-automation.com/2014/05/hard-vs-soft-assembly-automation-increased-factory-productivity/, [Online]; Consult: 2015-July-1

[4] John Paul MacDuffie, "From Fixed to Flexible: Automation and Work Organization trends", in Transforming Auto Assembly: International Experiences With Automation and Work organization, March 1996

[5] Ernie Hayden GICSP, "An Abbreviated History of Automation & Industrial Controls Systems", SANS institute, August 2014

[6] Stuart Bennett, "A brief History of Automatic Control", IEEE Control Systems, June 1996

[7] J. Clerk Maxwell, "On Governors", Proceedings of the Royal Society of London, January 1867

[8] Egyptian Water clock illustration. Available in http:// beforeitsnews.com/travel/2015/10/the-water-clock-of-the-ancient-egyptian-2475404.html, [Online]; Consult: 2015-July-1

[9] Boulton & Watt engine of 1788, digital image, Available in https:// en.wikipedia.org/wiki/Centrifugal_governor#/media/File:Boulton_and_Watt_centrifugal_governor-MJ.jpg, [Online]; Consult: 2015-July-1

[10] Original Modicon 084 PLC, digital image. Available in http://www.openautomation.de/detailseite/40-jahre-sps.html, [Online]; Consult: 2015-July-1

[11] Mario Hermann, "Design Principles for Industry 4.0 Scenarios", Business Engineering Institute, January 2015

[12] Pamela J. Waterman, "Manufacturing in the World of Industry 4.0", Available in http://www.deskeng.com/de/manufacturing-in-the-world-of-industrie-4-0/, [Online]; Consult: 2015-June-1

[13] Harald Hassenmuller, "The door The Recognized its body", Available in http://www.siemens.com/innovation/en/home/pictures-of-the-future/industry-and-automation/digital-factory-the-door-that-recognized-its-body.html, [Online]; Consult: 2015-July-1

[14] Arthur F. Pease, "Scenario 2030: Riding to Reality", Available in http://www.siemens.com/innovation/en/home/pictures-of-the-future/industry-and-automation/digital-factories-riding-to-reality.html, [Online]; Consult: 2015-July-1

[15] Susanne Gold, "Coming Soon: Personalized Factory Workstations", Available in http://www.siemens.com/innovation/en/home/pictures-of-the-future/industry-and-automation/digital-factories-personalized-workstations.html, [Online]; Consult: 2015-June-1

[16] EOS Lab, "Additive Manufacturing", Available in http://www.eos.info/additive_manufacturing/for_technology_interested, [Online]; Consult: 2015-July-1

[17] Tata McGraw, "Industrial Electronics and Control", Hill Education, 1998.

[18] Wojciech, "Personal comments on IEC61131-3 standard", Festo, July 2011.

[19] W.Bolton, "Programmable Logic Controllers", Elsevier Newnes, 2006.

[20] PLC cabinet, digital image. Available in http://www.njc-usa.com/wooddy/controls.htm, [Online]; Consult: 2015-June-3

[21] Jeff Payne, "Future of the PLC", Available in http://www.controleng.com/single-article/future-of-the-plc/a5e0a692be5b5a2f93dbe38215f770d1.html, Consult: 2015-July-4

[22] Sammy Natsui, “PLC Developments Increase Flexibility and Speed”. Available in http://www.controleng.com/single-article/plc-developments-increase-flexibility-and-speed/b5b891a11132d6622e32a617ba12c679 .html, [Online]; Consult: 2015-July-4

[23] Siemens, “PLC or DCS”. Available in http://w3.siemens.com/mcms/process-control-systems/SiteCollectionDocuments/efiles/pcs7/support/marktstudien/PLC_or_DCS.pdf, [Online]; Consult: 2015-July-6

[24] Tarun Agarwal, “Everything you need to know about DCS”. Available in http://www.elprocus.com/distributed-control-system-features-and-elements/, [Online]; Consult: 2015-July-6

[25] HIS Engineering, “DCS information”. Available in http://www.globalspec.com/learnmore/networking_communication_equipment/networking_equipment/distributed_control_systems_dcs

[26] Wikipedia, “Methods of production”, Available in https://en.wikipedia.org/wiki/Methods_of_production, [Online]; Consult: 2015-July-6

[27] Business Management Theory, “Production Methods”. Available in http://www.learnmanagement2.com/production.html, , [Online]; Consult: 2015-July-6

[28] Barry Young, “Recent Trends shape Future of DCS”. Available in http://www.automationworld.com/dcs/recent-trends-shape-future-distributed-control-systems, [Online]; Consult: 2015-July-6

[29] Brendan Galloway and Gerard Hancke, “Introduction to Industrial Control Networks”, IEEE, June 2012.

[30] S. Dijev, “Industrial Networks for Communication and Control”. Available in http://anp.tu-sofia.bg/djiev/PDF%20files/Industrial%20Networks.pdf, [Online]; Consult: 2015-July-12

[31] George Thomas, “Introduction to Modbus”. Available in http://www.ccontrols.com/pdf/Extv9n5.pdf, [Online]; Consult: 2015-July-13

[32] Marco Natale, “Understanding and using CAN”. Available in http://inst.cs.berkeley.edu/~ee249/fa08/Lectures/handout_canbus2.pdf, Consult: 2015-July-14

[33] Acromag, "Introduction to Profibus DP". Available in http://www.diit.unict.it/users/scava/dispense/II/Profibus.pdf/, [Online]; Consult: 2015-July-15

[34] Siemens, "Profinet Description". Available in http://www.siemens.fi/pool/products/industry/iadt_is/tuotteet/automaatiotekniikka/teollinen_tiedonsiirto/profinet/man_pnsystem_description.pdf, [Online]; Consult: 2015-July-16

[35] Darek Kominek, "OPC: The Ins and Outs to What it is About". Avaialble in http://www.automation.com/pdf_articles/Guide_to_OPC. pdf, [Online]; Consult: 2015-July-17

[36] K.J. Aström and T. Hägglund, "Automatic Tuning of PID Controllers", Instrument Society of America, 1988.

[37] Desborough L. & Harris, T. J., "Performance assessment measures for univariate feedback control". Canadian Journal of Chemical, 1992.

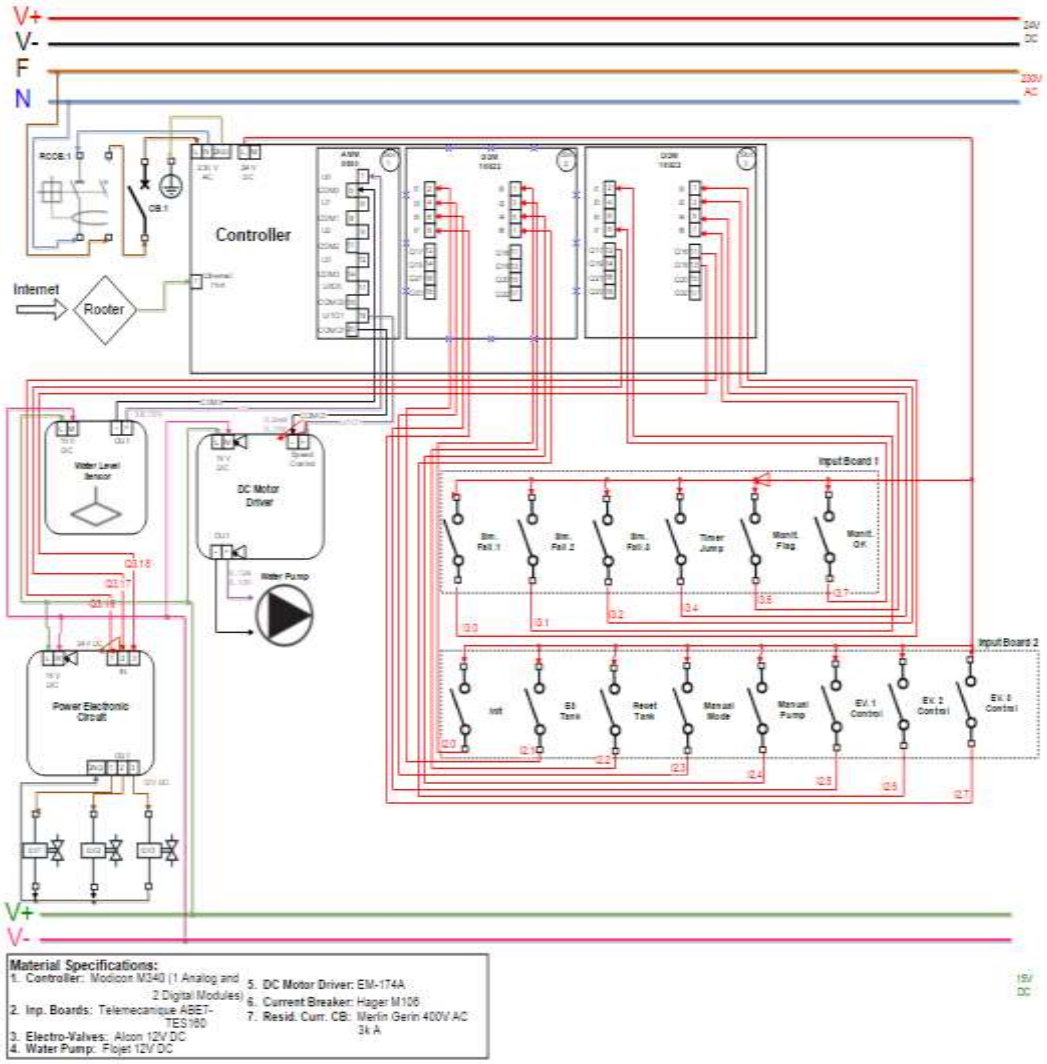
[38] Fspanero, "Análise de Componente Principais – PCA". Available in http : / / fspanero .wordpress.com/, [Online],Consult: 2015-August-20.

[39] Jolliffe, "Principal component analysis". New York: Springer-Verlag, ,1986.

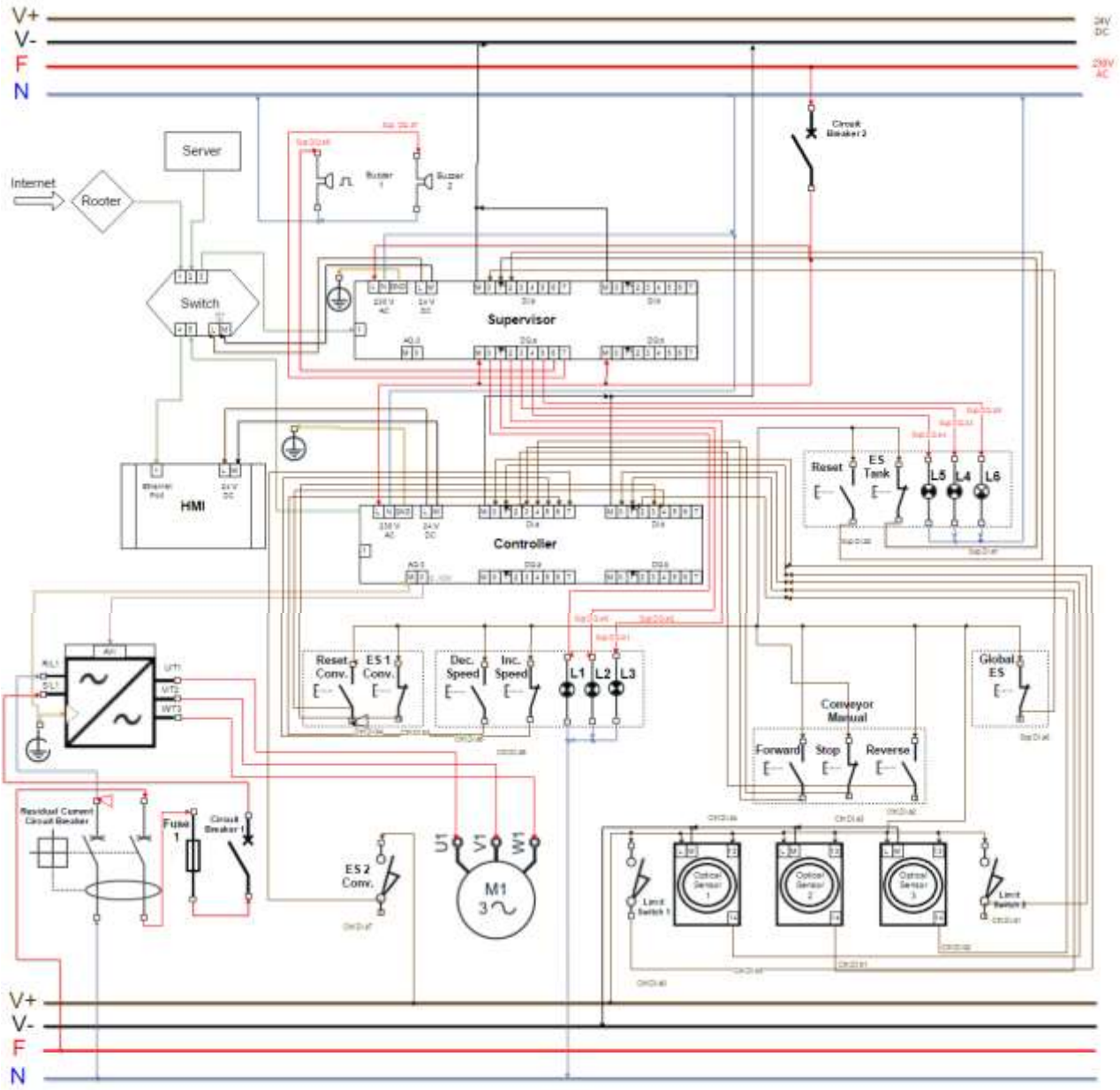
Appendix

A. Connection Schemes

- **Tank Panel Connection Scheme**:



- **Conveyor connection scheme panel**:



Material Specifications:

1. **Supervisor:** Siemens s7-1200
2. **Controller:** Siemens s7-1200
3. **HMI:** Siemens KTP600 Basic color PN
4. **Switch:** Siemens Simatic Net CSM1277
5. **Inverter:** Delta VFD-L
6. **Motor:** Efacec 3 phase AC Motor 120W (BF7 63M24)
7. **Resid. Curr. CB:** Merlin Gerin 400V AC 3k A
8. **Current Breaker:** Hager M106
9. **Optical Sensor:** Sick Optic 922F
10. **L. Switches:** Telemecanique XCK-M
11. **Buzzers:** Zumbo 230V AC
12. **24V Power Source:** Velleman PS613

B. I/O Tables

- **Supervisor (Simatic s7-1200)**:

Table 5- Supervisor used Inputs

Name	Type	Address	Mode	Notes
Global ES_a	Digital Relay of 24 V DC	DI.a 0 / %I0.0	NC	
ES Process2_a	Digital Relay of 24 V DC	DI.a 1 / %I0.1	NC	
Reset_proc2	Digital Relay of 24 V DC	DI.a 2 / %I0.2	NO	-Returns system to initial state; -Unblocks system from Emergency stop.

Table 6- Supervisor used outputs

Name	Type	Address	Notes
L1	Digital Relay of 230 V AC	DQ.a 0 / %Q0.0	-Indicates that the conveyor is active
L2	Digital Relay of 230 V AC	DQ.a 1 / %Q0.1	-Indicates that the Tank is active
L3	Digital Relay of 230 V AC	DQ.a 2 / %Q0.2	-Indicates that the supervisor is active
L4	Digital Relay of 230 V AC	DQ.a 3 / %Q0.3	-Indicates process2 I stopped.
L5	Digital Relay of 230 V AC	DQ.a 4 / %Q0.4	--Indicates process1 is stopped.
L6	Digital Relay of 230 V AC	DQ.a 5 / %Q0.5	-System mode;
Buzzer1	Digital Relay of 230 V AC	DQ.a 6 / %Q0.6	-Indicates Failure on process1.
Buzzer2	Digital Relay of 230 V AC	DQ.a 7 / %Q0.7	-Indicates Failure on process2.

- **Conveyor controller (Simatic s7-1200)**:

Table 7-Conveyor controller used inputs

Name	Typo	Address	Mode	Notes
Limit Switch 1	Digital Relay of 24 V DC	DI.a 0 / %I0.0	NO	
Limit Switch 2	Digital Relay of 24 V DC	DI.a 1 / %I0.1	NO	

Motor_Forward	Digital Relay of 24 V DC	DI.a 4 / %I0.4	NO	-Manual mode feature; -Secondary button, activated via pressing/timer.
Reset Process 1 a	Digital Relay of 24 V DC	DI.a 4 / %I0.4	NO	-Secondary button, activated via pressing/timer;
Motor_Stop	Digital Relay of 24 V DC	DI.a 3 / %I0.3	NC	-Manual mode feature;
Manual mode Flag	Digital Relay of 24 V DC	DI.a 3 / %I0.3	NC	-Secondary button, activated via pressing/timer; -Forces system to activate manual mode;
-Motor_Reverse	Digital Relay of 24 V DC	DI.a 2 / %I0.2	NO	-Manual mode feature;
Init	Digital Relay of 24 V DC	DI.a 2 / %I0.2	NO	-Secondary button, activated via pressing/time. -Initiates conveyor control system;
Inc frequency	Digital Relay of 24 V DC	di.a.5 / %I0.5	NC	-Manual mode feature;
Failure monitor flag	Digital Relay of 24 V DC	di.a.5 / %I0.5	NC	-Secondary button, activated via pressing/timer; -Tells the system, a technician is present during the occurrence of the failure.
Dec frequency	Digital Relay of 24 V DC	di.a6 / %I0.6	NO	-Manual mode feature;
Monitor Ok flag	Digital Relay of 24 V DC	di.a6 / %I0.6	NO	-Secondary button, activated via pressing/timer; -Tells the system, the failure was fixed.
ES Process1 a	Digital Relay of 24 V DC	DI.a 7 / %I0.7	NO	-User's contact protection for process1.
ES Process1 b	Digital Relay of 24 V DC	DI.b 3 / %I1.3	NC	
Reset Process1 b	Digital Relay of 24 V DC	DI.b 4 / %I1.4	No	
Failure_proc1	Digital Relay of 24 V DC	DI.b 4 / %I1.4	No	-Secondary button, activated via pressing/timer; -Simulated failure of the conveyor.

Optical Sensor 1	Digital Relay of 24 V DC	DI.b 0 / %I1.0	NO	-There is the need to dock an aluminum reflector plate onto the metal part carrier.
Optical Sensor 2	Digital Relay of 24 V DC	DI.b 1 / %I1.1	NO	""
Optical Sensor 3	Digital Relay of 24 V DC	DI.b 2 / %I1.2	NO	""

Table 8- Conveyor controller used inputs

Nome	Type	Address	Notes
v/f Control of the motor	Analogic signal of 0 to 10 V DC	AQ 0 / %Qw80	F-max=40 Hz=27648; F-stop=0 Hz=16800; F-min=-40 Hz=5600.

- **Tank controller (Modicon M340)**:

Table 9- Tank controller used inputs

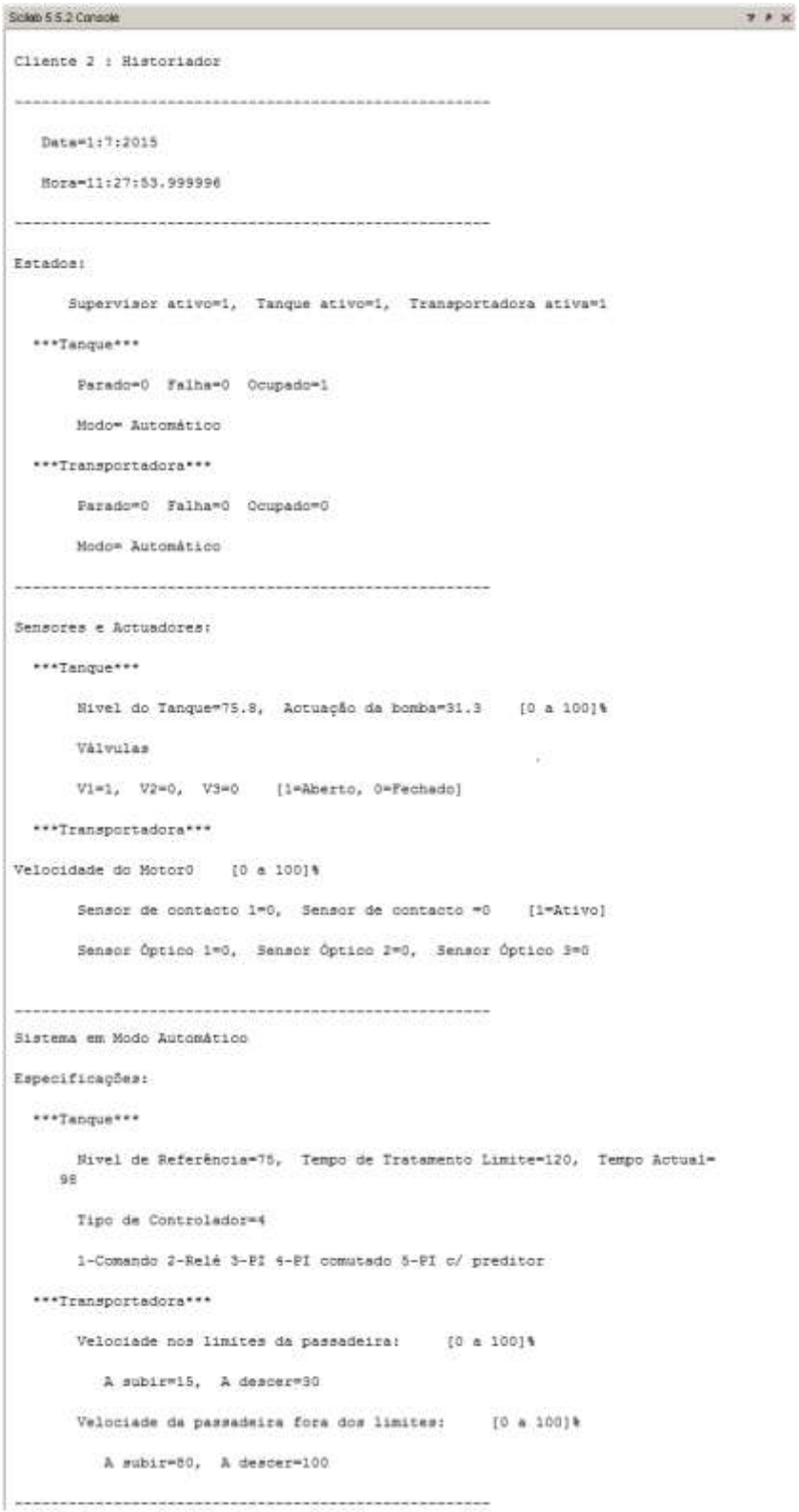
Name	Type	Address	Mode	Notes
Init	Digital Solid State input of 24 V DC	%I0.2.0		-Initiates tank control system;
ES process2_b	Digital Solid State input of 24 V DC	%I0.2.1		
Reset_b	Digital Solid State input of 24 V DC	%I0.2.2		
Manual_mode	Digital Solid State input of 24 V DC	%I0.2.3		-Flag that forces system into manual mode.
Manual_pump	Digital Solid State input of 24 V DC	%I0.2.4		-If on, activates the water pump; -Default voltage applied to water pump= 4V
Electro Valve 1 control	Digital Solid State input of 24 V DC	%I0.2.5		
Electro Valve 2 control	Digital Solid State input of 24 V DC	%I0.2.6		
Electro Valve 3 control	Digital Solid State input of 24 V DC	%I0.2.7		
Simulated Failure 1	Digital Solid State input of 24 V DC	%I0.3.0		-Activates Failure flag;(major failure)

Simulated Failure 2	Digital Solid State input of 24 V DC	%I0.3.1		-Opens Electro-valve 3 (minor failure)
Simulated Failure 3	Digital Solid State input of 24 V DC	%I0.3.2		-Opens Electro-valve 2 and 3 (mid failure)
Timer Jump	Digital Solid State input of 24 V DC	%I0.3.4		-Ignores the wait time associated to treatment of the current metal part on tank.
Monitoring Flag	Digital Solid State input of 24 V DC	%I0.3.6		-Flag used to notify the system that a user is monitoring the system, in case of failure.
Monitor Ok Flag	Digital Solid State input of 24 V DC	%I0.3.7		-Tells the system, the failure was fixed.
Tank Level Sensor	Analogic signal of 0 to 10 V DC	%Iw0.1.0 (U0/com0)		0%= 3830; 25%=4350; 50%=5300; 75%=6760; 100%=9320.

Table 10- Tank controller used outputs

Name	Type	Address	Notes
Electro Valve 1 actuator	Digital Solid State output of 24 V DC	%Q0.3.16	
Electro Valve 2 actuator	Digital Solid State output of 24 V DC	%Q0.3.17	
Electro Valve 3 actuator	Digital Solid State output of 24 V DC	%Q0.3.18	
Tank's Water Inflow Pump	Analogic signal +/- 10V DC	%Qw0.1.5 (U101/Com101)	Voltage range specified= [0 to 8]V

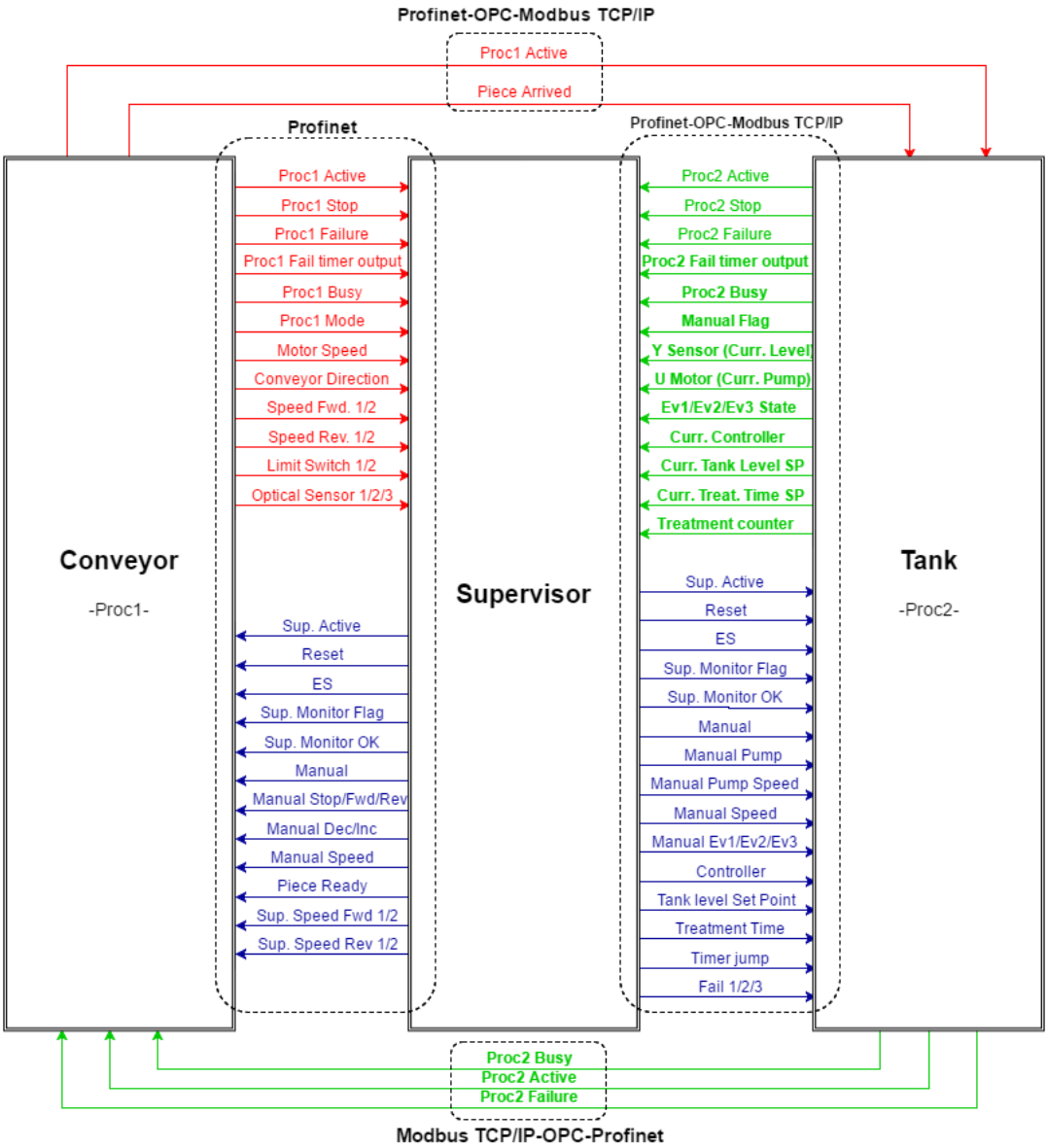
C. Operational Historian Log



```
Scilab 5.5.2 Console

Cliente 2 : Historiador

-----------------------------------------------------

   Data=1:7:2015

   Hora=11:27:59.999996

-----------------------------------------------------

Estados:

      Supervisor ativo=1,  Tanque ativo=1,  Transportadora ativa=1

  ***Tanque***

       Parado=0  Falha=0  Ocupado=1

       Modo= Automático

  ***Transportadora***

       Parado=0  Falha=0  Ocupado=0

       Modo= Automático

-----------------------------------------------------

Sensores e Actuadores:

  ***Tanque***

       Nivel do Tanque=75.8,  Actuação da bomba=31.3    [0 a 100]%

       Válvulas

       V1=1,  V2=0,  V3=0    [1=Aberto, 0=Fechado]

  ***Transportadora***

Velocidade do Motor0    [0 a 100]%

       Sensor de contacto 1=0,  Sensor de contacto =0    [1=Ativo]

       Sensor Óptico 1=0,  Sensor Óptico 2=0,  Sensor Óptico 3=0


-----------------------------------------------------

Sistema em Modo Automático

Especificações:

  ***Tanque***

       Nivel de Referência=75,  Tempo de Tratamento Limite=120,  Tempo Actual=
     98

       Tipo de Controlador=4

       1-Comando 2-Relé 3-PI 4-PI comutado 5-PI c/ preditor

  ***Transportadora***

       Velociade nos limites da passadeira:     [0 a 100]%

          A subir=15,  A descer=30

       Velociade da passadeira fora dos limites:     [0 a 100]%

          A subir=80,  A descer=100

-----------------------------------------------------
```

D. Communication between Components

- *Information traded between Supervisor and the Processes:*



- *Information traded between Supervisor and the Remote Clients:*